

UNCLASSIFIED

AD NUMBER

ADB019079

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to U.S. Gov't. agencies only; Test and Evaluation; NOV 1976. Other requests shall be referred to Air Force Armament Laboratory, Attn: DLMM, Eglin AFB, FL 32542.

AUTHORITY

USADTC ltr, 23 Oct 1979

THIS PAGE IS UNCLASSIFIED

THIS REPORT HAS BEEN DELIMITED  
AND CLEARED FOR PUBLIC RELEASE  
UNDER DOD DIRECTIVE 5200.20 AND  
NO RESTRICTIONS ARE IMPOSED UPON  
ITS USE AND DISCLOSURE.

DISTRIBUTION STATEMENT A

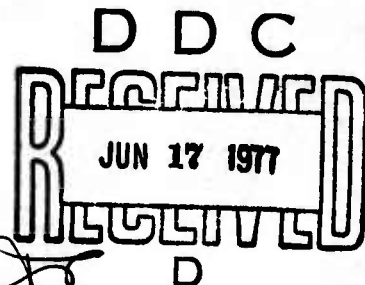
APPROVED FOR PUBLIC RELEASE;  
DISTRIBUTION UNLIMITED.

AFATL-TR-76-132, VOLUME I

**DIGITAL GUIDED WEAPONS TECHNOLOGY  
VOLUME I:  
DIGITAL PROCESSOR SYSTEM STUDIES**

**MISSILE SYSTEMS DIVISION  
HUGHES AIRCRAFT COMPANY  
CANOGA PARK, CALIFORNIA 91304**

**NOVEMBER 1976**



**FINAL REPORT: AUGUST 1974-NOVEMBER 1976**

Distribution limited to U. S. Government agencies only; this report documents test and evaluation; distribution limitation applied November 1976. Other requests for this document must be referred to the Air Force Armament Laboratory (DLMM), Eglin Air Force Base, Florida 32542.

**AIR FORCE ARMAMENT LABORATORY**

**AIR FORCE SYSTEMS COMMAND • UNITED STATES AIR FORCE**

**EGLIN AIR FORCE BASE, FLORIDA**



AD B019079

AD No. \_\_\_\_\_  
DDC FILE COPY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFATL TR-76-132, Volume 1	2. GOVT ACCESSION NO. N/A	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DIGITAL GUIDED WEAPONS TECHNOLOGY, VOLUME I. DIGITAL PROCESSOR SYSTEM STUDIES.		5. TYPE OF REPORT & PERIOD COVERED Final Report. Aug 1974 - Nov 1976
7. AUTHOR(S) F. W. Hardy, S. Y. Wong R. L. Hoolko A. J. Roach R. L. Woolley		6. PERFORMING ORG. REPORT NUMBER DGWT-0230-1
9. PERFORMING ORGANIZATION NAME AND ADDRESS Missile Systems Division Hughes Aircraft Company Canoga Park, CA., 91304		8. CONTRACT OR GRANT NUMBER(s) F08635-75-C-0014
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Armament Laboratory Armament Development and Test Center Eglin Air Force Base, Florida 32542		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project No. 670B Task No. 105 Work Unit No. 001
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE Nov 1976
		13. NUMBER OF PAGES 163
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution limited to U. S. Government agencies only; this report documents test and evaluation; distribution limitation applied November 1976. Other requests for this document must be referred to the Air Force Armament Laboratory (DLMM), Eglin Air Force Base, Florida 32542.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Available in DDC.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Modular weapon Integration Digital Processor Multiplexed Digital Bus		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)  A digital processing system is defined which will support the integration of the components of an advanced modular weapon system. Modularity requirements of the weapon system led to the choice of a multiplexed digital bus as the prime communication channel in the weapon. The weapon bus presents a common interface to all sub- over		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)




UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

20. ABSTRACT (Concluded)

systems and provides a common functional interface between weapon sections. The combination of a digital processor and the weapon bus forms the integration subsystem of the weapon. The functions which are performed in the digital processor to accomplish integration are called the CORE functions. These CORE functions are system management, flight control, and the strapdown inertial navigation function. System management includes weapon system identification, communication control, and self-test. It provides the real time control of the weapon system processor. The strapdown inertial reference function includes the data filtering and position update tasks which interface midcourse guidance sensors with the inertial navigation function.

The throughput requirement for the digital processing system is determined by time-critical functions in the flight control subsystem. Propagation delay restrictions in processing inertial sensor data for the flight control stabilization function set both bus transmission rate and processor functional throughput.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

PREFACE

2

This report was prepared by the Missile Systems Division, Hughes Aircraft Company, Canoga Park, California 91304, under Contract No. F08635-75-C-0014 with the Air Force Armament Laboratory, Armament Development and Test Center, Eglin Air Force Base, Florida. Major Robert L. Haney (DLMM) monitored the program for the Armament Laboratory. This effort was conducted during the period from August 1974 to November 1976.

This report consists of three volumes. Volume I contains Digital Processor System Studies. Volume II is concerned with System Simulations. Volume III deals with Programmable Digital Autopilot. This is Volume I.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER:

*Clifford H. Allen, Jr.*  
CLIFFORD H. ALLEN, JR., Colonel, USAF  
Chief, Guided Weapons Division

ACCESSION 347	
NTIS	White Section <input type="checkbox"/>
DDC	Buff Section <input checked="" type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
SPECIAL	

13

DDC  
RECEIVED  
JUN 17 1977  
D

## TABLE OF CONTENTS

Section	Title	Page
I.	INTRODUCTION	1
II.	WEAPON SYSTEM DEFINITION	5
III.	DIGITAL SYSTEM DESIGN PROCEDURE	11
3.1	System Design Process	11
3.2	Subsystem Characterization	12
3.3	Digital Function Partitioning	13
3.4	Digital Processor Design	15
IV.	SUBSYSTEM DESCRIPTIONS AND REQUIREMENTS	17
4.1	Midcourse Guidance Subsystems	17
4.1.1	Strapdown Inertial Reference Subsystem	17
4.1.2	Radiometric Area Correlation Subsystem	21
4.1.3	LORAN Subsystem	23
4.1.4	E-O Data Link Subsystem	27
4.1.5	TERCOM Subsystem	29
4.1.6	DME Subsystem	30
4.1.7	Global Positioning System (GPS) Subsystem	30
4.2	Terminal Guidance Subsystems	32
4.2.1	Electro-Optical and Imaging Infrared Seeker Subsystems	33
4.2.2	Radiometric Contrast Seeker Subsystem	34
4.2.3	Semiactive Laser Seeker Subsystem	39
4.2.4	Anti-Radiation Seeker Subsystem	39
4.3	Flight Control Subsystem	41
4.4	Armament Subsystem	42
4.5	Aircraft Interface	42
V.	POINT DESIGN FOR WEAPON CONFIGURATIONS	45
5.1	Design Ground Rules for Point Designs	45
5.2	Weapon Configuration Point Design	46
5.2.1	Weapon Configuration I	46
5.2.2	Weapon Configuration II	48
5.2.3	Weapon Configuration III	50
5.2.4	Configuration Requirements Summary	50
5.3	Integration in Point Designs	52
VI.	THE MODULAR WEAPON AND INTEGRATION	55
6.1	Modular Weapon Characteristics	55
6.1.1	Description of Modular Weapon (Weapon Assembly Level)	55
6.1.2	Growth Considerations	57

## TABLE OF CONTENTS (Continued)

Section	Title	Page
6.2	The Integration Subsystem	57
6.3	Processor Functions Required for Integration	59
6.3.1	System Management Functions	59
6.3.2	Flight Control	61
6.3.3	Strapdown Inertial Reference	61
6.3.4	CORE Functions	61
6.3.5	Bus Transmission Requirements for CORE Functions	61
6.4	CORE Digital Processing System	62
VII.	DIGITAL PROCESSING SYSTEM DESIGN	63
7.1	Weapon Bus Design	63
7.1.1	Bus Topology and Control Structure	64
7.1.2	Bus Message Word Length	69
7.1.3	Weapon Bus Tradeoff Summary	73
7.2	Processor System Architecture	73
7.2.1	Flight Control Function in Separate Processor	74
7.2.2	CORE Function Processing Requirements	82
7.2.3	Strapdown Inertial Reference in Separate Processor	83
7.2.4	Summary	84
7.3	Software Design	84
7.4	Digital Processor Architecture	90
7.5	Digital Processing System Requirements	93
7.5.1	Processor Throughput Requirements	93
7.5.2	Processor Program Memory Requirements	96
7.5.3	Processor Operand Memory Requirements	99
7.5.4	Weapon Bus Requirements	99
7.6	Configuration Dependent Processing Functions	100
7.6.1	Weapon Configuration I	100
7.6.2	Weapon Configuration II	101
7.6.3	Weapon Configuration III	102
7.6.4	Generic Classes of Configuration Dependent Functions	103
7.6.5	Weapon System Considerations	103
VIII.	TECHNOLOGY STUDY	105
8.1	DP-X Design Study	106
8.1.1	Design Approach	106
8.1.2	Instruction Formats	107
8.1.3	Typical Instructions	109
8.1.4	DP-X Units and Buses	109
8.1.5	DP-X Pipeline	109
8.1.6	Program Control	112
8.1.7	Comparison of DP-1 and DP-X Speeds	114
8.1.8	DP-X Packaging	114
8.1.9	Summary	116

## TABLE OF CONTENTS (Continued)

Section	Title	Page
8.2	Low Power Schottky Cell Array Technology	118
8.3	SOS C-MOS LSI Technology	119
8.4	I <sup>2</sup> L Technology	121
8.5	Technology Conclusions	124
IX.	COST ANALYSIS	125
9.1	Introduction	125
9.2	Selection of Processor Types for Cost Analysis	127
9.3	Configuration Descriptions	129
9.3.1	Overall Description	129
9.3.2	Box Descriptions	129
9.3.3	Circuit Card Descriptions	130
9.4	Production Costs	136
9.4.1	Cost Model	137
9.4.2	Development of Factory Cost	137
9.4.3	Prime Material Costs	139
9.4.4	Labor Costs	143
9.4.5	Production Cost Comparisons	143
9.5	Development Costs	152
9.5.1	Development Cost Model	152
9.5.2	Risk Assessment	156
9.5.3	Software Considerations	157

## LIST OF FIGURES

Figure	Title	Page
1	Digital Processor Facilitates Modular Weapon System Integration	6
2	A Weapon Configuration is a Set of Functional Modules	6
3	Candidate Weapon Configuration No. 1 (High Complexity)	8
4	Candidate Weapon Configuration No. 2 (Moderate Complexity)	8
5	Candidate Weapon Configuration No. 3 (Low Complexity)	9
6	System Design Process	12
7	Subsystem Characterization	13
8	Digital Function Partitioning	14
9	Digital Processor Design	16
10	Midcourse Guidance Characterization	18
11	Strapdown Inertial Reference Block Diagram	18
12	Inertial Reference Digital Processing Functions	20
13	Radiometric Area Correlator Block Diagram	22
14	Correlation Algorithms	25
15	LORAN Subsystem Block Diagram	27
16	Electro-Optical Data Link Subsystem Block Diagram	28
17	TERCOM Subsystem Block Diagram	29
18	GPS Subsystem Functional Flow Diagram	31
19	Principal GPS Guidance Processing Functions	32
20	Terminal Guidance Characterization	33
21	EO/IIR Seeker Block Diagram	34
22	ATVS Tracking Algorithm	35
23	Optical Tracker Algorithm	37
24	Processing Requirements – EO/IIR Seeker	38
25	Radiometric Contrast Seeker Block Diagram	38
26	Semiactive Laser Seeker Block Diagram	40
27	Anti-Radiation Seeker Block Diagram	40
28	Flight Control Subsystem Block Diagram	41
29	Configuration I Functional Block Diagram	47
30	Throughput Requirements – Configuration I	48

# LIST OF FIGURES (Continued)

Figure	Title	Page
31	Configuration II Functional Block Diagram	49
32	Throughput Requirements - Configuration II	50
33	Configuration III Functional Block Diagram	51
34	Throughput Requirements - Configuration III	52
35	Digital Processing System Configuration	62
36	Weapon Bus Topology	65
37	Rotating Window Control Structure	67
38	Bus Race Control Structure	67
39	Central Control Structure	68
40	Message Identification Coding	70
41	Bus Message Formats	72
42	Interrupt Bus Word Format	72
43	Processing Sequence for Digital Functions	75
44	Distributed Processing Configuration Examples - Flight Control Functions	75
45	Flight Control Function Sequences	77
46	Stabilization Function Operations	79
47	Throughput versus Bus Rate	80
48	Direct Interconnection of Program Modules	86
49	Program Structure Using an Executive	86
50	Executive Routines	87
51	Executive Routines and Tables	89
52	Peak Throughputs for Time-Critical Flight Control Stabilization Function	95
53	IBM and Interdata Formats	108
54	DP-X Instruction Format Includes Interdata or IBM Format	108
55	Typical Instructions	110
56	DP-X Units and Buses	110
57	DP-X Pipeline	111
58	Normal Cycle	112
59	Program Control	113
60	Address Processor (ALU-X)	114
61	Comparison of DP-I and DP-X Speeds	115
62	Advanced DP Address Processor	116

## LIST OF FIGURES (Continued)

Figure	Title	Page
63	Advanced DP	117
64	LSI Cell Array	120
65	SOS-CMOS (SOCM) Processor Internal and External Elements	121
66	MCU Chip	122
67	Microprogram Control Chip	123
68	Selection of Processor Types for Cost Tradeoffs	128
69	Cost History for Military Memories	141



## LIST OF TABLES

Table	Title	Page
1	Operations and Instruction Types	15
2	Processing Requirements – Strapdown Inertial Reference	21
3	Processing Requirements – RAC Subsystem	23
4	Processing Requirements – LORAN Subsystem	28
5	Processing Requirements – EO Data Link	28
6	Processing Requirements – TERCOM Subsystem	30
7	GPS Guidance Subsystem Digital Processing Requirements	31
8	Processing Requirements – Antiradiation Seeker	41
9	Processing Requirements – Flight Control	42
10	Weapon Configurations	46
11	Configuration Processing Requirements Summary	52
12	Interface Signals for Point Designs	53
13	Bus Transmission Requirements for Fixed-Design Weapon Configurations	62
14	Bus Transfer Efficiency	72
15	Instructions Executed/Iteration	78
16	System Operations Required for Each Stabilization Computation	78
17	CORE Function Processing Requirements	83
18	Instruction Categories	91
19	Estimated Increase in Program Memory Requirements	98
20	DP Requirements for RAC Subsystem Functions	101
21	DP Requirements for LORAN, Data Link, EO Seeker Functions	102
22	DP Requirements for Data Link, IIR Seeker Functions	102
23	Allocation of Data Memory Address Space	133
24	Data Memory Allocation for Configuration 5	135
25	Memory Arrangement for Configuration 6	135
26	Parts List for Processor Input/Output	136
27	Estimated Costs for Integrated Circuits in 1980	140
28	Projected Costs for LSI Monoprocessor CPU	142
29	Number of Chips of Each Type Used in Configuration 1	144
30	Number of Chips of Each Type Used in Configuration 2	145

# LIST OF TABLES (Continued)

Table	Title	Page
31	Number of Chips of Each Type Used in Configuration 3	146
32	Number of Chips of Each Type Used in Configuration 4	147
33	Number of Chips of Each Type Used in Configuration 5	148
34	Number of Chips of Each Type Used in Configuration 6	149
35	Total Material Costs	150
36	Cumulative Average Cost per Unit for 5000 Units	150
37	Program Cost Summary for Configuration 1	153
38	Hardware Development Cost Summary (in \$1000)	153
39	Input Data for Estimating Hardware Development Costs for Configurations 1, 2 and 3	154
40	Input Data for Estimating Hardware Development Costs for Configurations 4, 5 and 6	155

## SECTION I

### INTRODUCTION

The Digital Guided Weapon Technology program was initiated with the general intent of determining the role of digital processing techniques in guided weapons. Recognizing that too broad a scope can be self-defeating for this kind of program, the Air Force set two specific goals to be accomplished. The first was a near term application of digital processing to an existing weapon family. Specifically, a digital autopilot for the GBU-15 weapon was to be designed and evaluated. The evaluation required fabrication of a brassboard digital processor for the autopilot and verification of its performance with a hybrid simulation. This part of the program was completed in 1975. Based on the results of that effort, a separate program was initiated to bring the GBU-15 digital autopilot into engineering development. The Program-mable Digital Autopilot (PDAP) work is reported in a separate volume.

The second goal was to determine how digital processing techniques could be used to assist in integrating the components of an advanced modular weapon system. Earlier studies sponsored by the Air Force had investigated the characteristics of a modular weapon. The results of one of these, as expressed in AFATL-TR-72-202, were used as a starting point to define the weapon system to be studied in the DGWT program. Specifically, the program was to accomplish the following:

1. Determine what functions could be done digitally in the weapon system.
2. Determine what the digital processing system should do to assist in integrating the weapon components.
3. Determine what other functions should be done in the digital processor.
4. Define the interface of the digital processing subsystem within the weapon system.
5. Determine the requirements of the digital processing system.
6. Produce a preliminary design of the digital processing system.
7. Build two breadboard processing systems.
8. Evaluate the breadboard systems in hybrid simulations and other tests.

The present volume reports on the first six of the tasks listed above. The description of the breadboard hardware and the results of system evaluation are reported in separate volumes. This volume is organized so as to present a step-by-step accounting of the design process which ended with the preliminary design of the digital processing system presented in Section VII. The performance requirements for the processing system are summarized in Section VII and are presented in greater detail in a specification which is published as a separate document.

The major findings of the study are summarized below along with a brief description of the contents of other sections in this book.

Modularity requirements of the weapon system led to the choice of a multiplexed digital bus as the prime communication channel in the weapon. The weapon bus presents a common interface to all subsystems and provides a common functional interface between weapon sections. The combination of a digital processor and the weapon bus form the integration subsystem of the weapon. The functions which are performed in the digital processor to accomplish integration are called the CORE functions. These CORE functions are system management, flight control, and the strapdown inertial navigation function. System management includes weapon system identification, communication control, and self-test. It provides the real time control of the weapon system processor. The strapdown inertial reference function includes the data filtering and position update tasks which interface midcourse guidance sensors with the inertial navigation function.

Other weapon functions can be performed in the digital processor. These additional functions are not typically common to most weapon configurations as are the CORE functions. The criteria for adding configuration dependent functions to the digital processor repertoire are:

- No increase in digital processing system requirements.
- There should be an economic benefit.
- Subsystem interfaces are simplified or at least not complicated.

Some functions which are good subjects for digital implementation do not satisfy the above criteria. An example is the video tracker in the electro-optical (E-O) seeker. Performing this function in the integration subsystem does increase bus transmission requirements and processor throughput requirements. Moreover, it complicates the subsystem interface. While such functions should be performed digitally, it should be in a separate processor.

The throughput requirement on the digital processing system is determined by time critical functions in the flight control subsystems. Propagation delay restrictions in processing inertial sensor data for the flight control stabilization function set both bus transmission rate and processor throughput capability.

The digital processor itself could be implemented either as a single processor or a distributed processor. Tradeoff studies showed no system or economic advantage to using a distributed processing system. In fact, for the CORE functions, using a distributed processing system leads to higher costs than using a single processor.

The software structure recommended for the digital processing system includes a simple executive which provides the interface between the hardware system and the software system, and also

provides the interface between functional software modules. This structure is consistent with weapon modularity requirements. Software modules can be added to or deleted from the system with little or no effect on other modules.

The weapon system used as the basis for the study is described in Section II.

Three weapon configurations are selected for detail study. The configurations were chosen to cover a range of complexities so that the effect of varying requirements on the digital processor could be assessed.

The procedures used to carry out the tradeoff studies and arrive at a digital system design are discussed in Section III. Design methodology and tradeoff criteria are also described here.

Weapon subsystems is the subject of Section IV. A selection of major subsystems is chosen, and the digital processing requirements for each are analyzed. The viewpoint here is to determine what functions in the subsystem can be done digitally and what is the effect on processor requirements by choosing different interfaces. The digital processor requirements for each subsystem are determined as a function of the interface.

The design process is carried to the system level in Section I where the requirements for each of the three chosen weapon configurations are examined. The viewpoint of this section is to determine the requirements of a digital processor that is optimized for a single weapon configuration. Subsystem interfaces are selected and, for each configuration, the requirements corresponding to those interfaces are selected. It is made clear that optimizing for particular configurations does not lead to a common set of requirements for the digital processor.

The role of the digital processor in integrating the components of a fixed-design weapon is discussed.

The question of integration in a modular weapon is addressed in Section VI. The method of integration in a fixed-design weapon must be extended to satisfy modularity requirements. The requirements lead to the selection of a multiplexed digital bus and a digital processor to form the integration subsystem. Weapon functions which are closely related to the integration process should be performed in the digital processor. These CORE functions are selected.

The digital processing system design is described in Section VII. Weapon bus tradeoffs lead to the selection of the bus configuration. A tradeoff of processor configurations leads to the selection of a single processor to perform the CORE functions. The software structure is described and performance requirements on the digital processor system are presented. The section closes with a brief discussion of other functions, other than CORE functions, which might be performed in the digital processor.

Section VIII contains the results of a study conducted to determine appropriate technology for implementing the digital processor.

The cost analysis reported in Section IX analyzed cost differences in the digital processing system which resulted from different ways of implementing the processor. The tradeoff contains a comparison of single processor systems and distributed processor systems.

## SECTION II

### WEAPON SYSTEM DEFINITION

Different tactical air-to-surface guided weapons share many common functions. Figure 1 illustrates a generic guided weapon configuration broken into its basic functional parts. The commonality of these parts can be clearly seen in existing weapons. Each weapon, for example, requires some form of guidance to indicate the direction of the intended target, a control module to produce maneuvers to reach the target, and a warhead module to destroy the target upon impact.

In recent years, it has been widely recognized that there are significant benefits to approaching tactical weapon design from the viewpoint of modularity. The chief benefit is reduced cost, both in weapon system development and production. This cost benefit comes principally through commonality of design. For example, although it may be attractive to consider different types of weapon guidance under particular conditions (e.g., maximum accuracy E-O guided weapons in good weather and lower accuracy weapon guidance such as DME for area targets) the same airframe and control modules may be entirely adequate for either condition. Therefore, development of a weapon design which utilizes different guidance modules for the same airframe would be cost effective. Although some small additional development cost would be incurred to ensure interchangeability of the guidance units, development cost associated with two different airframes and control modules would be avoided. In production, the costs can be reduced since the fabrication of any number of a single design is substantially less costly than that of one-half the number of each of two separate designs.

While the benefits of modularity are evident from the preceding discussion, there have been a number of stumbling blocks in the way of successfully achieving it. One such stumbling block is providing the functional adaptivity required to accommodate the different modules. For example, a laser terminal seeker has somewhat different interface requirements than an E-O terminal seeker. The output steering signals have different filtering requirements. Different warheads require different guidance laws to give the optimum trajectory. Over the entire subsystem spectrum, there are many of these differences which require some medium to provide the proper adjustment. The adjustment can be made by adapter modules, either permanently placed in the subsystem or used in an ad hoc fashion. Either method has disadvantages. One of the prime objectives of this study was to determine how a digital processing system can help in obtaining this functional adaptivity.

As a first step in analyzing the problem, a basic weapon system was defined. The elements of this weapon system are illustrated in Figure 2.



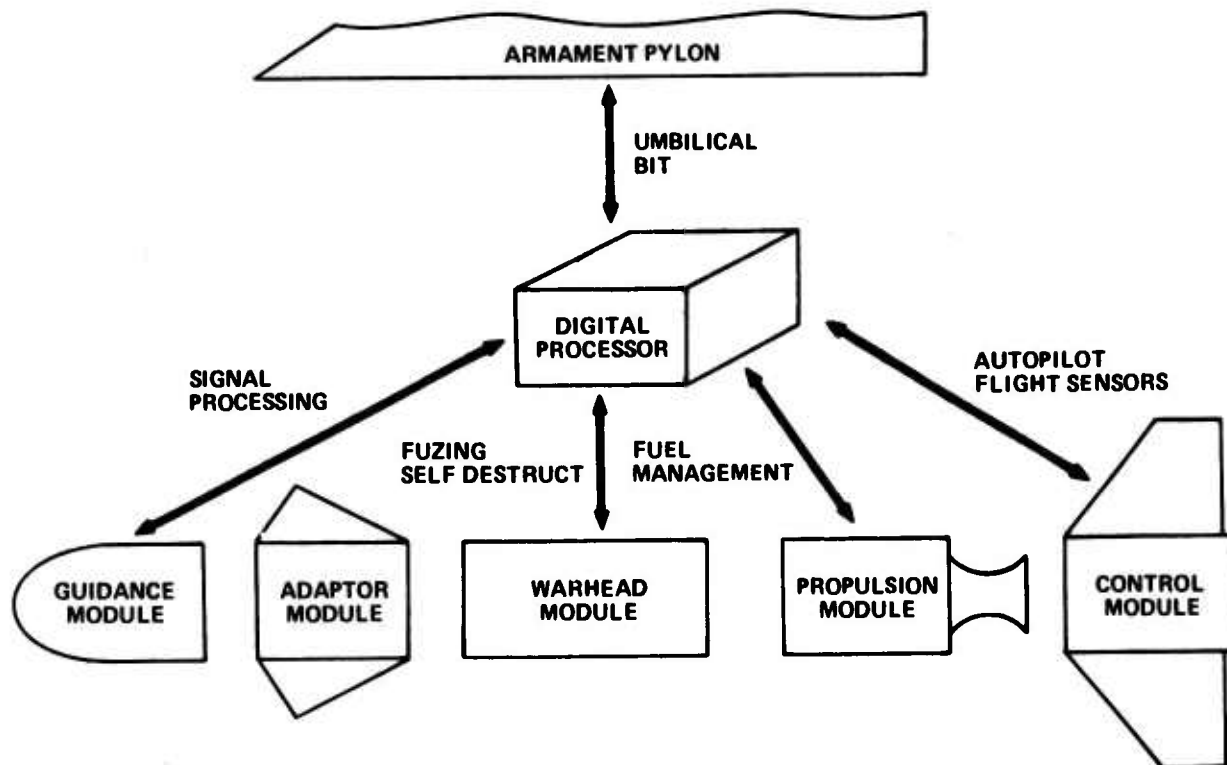


Figure 1. Digital Processor Facilitates Modular Weapon System Integration

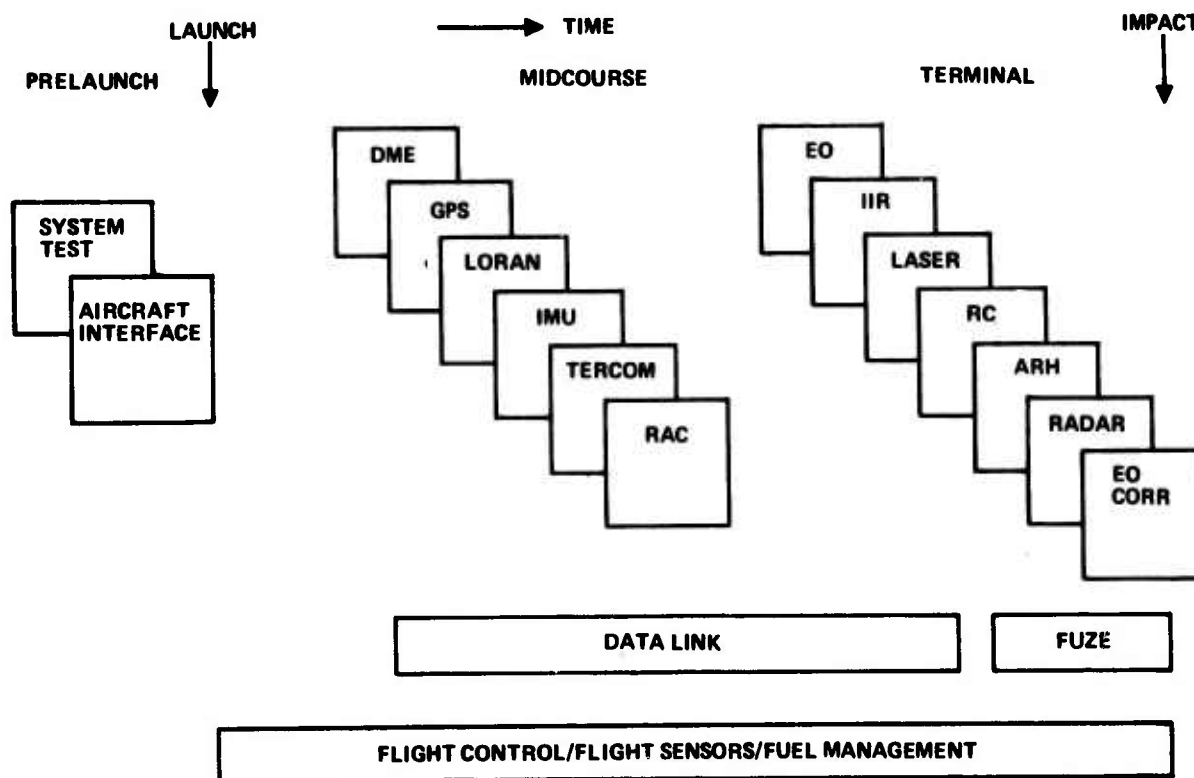


Figure 2. A Weapon Configuration is a Set of Functional Modules



The figure illustrates the candidate tactical weapon subsystem considered in establishing total weapon processing requirements. These are grouped in terms of primary functions such as midcourse and terminal guidance and arranged in an approximate time line of when they occur during missile flight. The weapon configuration is simply a combination of one of each of the functional elements. From these candidate subsystems, three weapon configurations have been selected for detailed analysis. The configurations have been specifically chosen to represent a range from low to high processing complexity to determine how weapon complexity affects digital processor requirements.

The most complex of the three selected weapon configurations is shown in Figure 3. The vehicle is a low volume ramjet type of cruise missile, such as is being developed at the Naval Weapons Center, China Lake, California, having a range on the order of a few hundred miles. A fuel management processing function is assumed for the vehicle's engine.

Guidance for the vehicle is provided during both midcourse and terminal phases of flight by navigation in a guidance computer, using the outputs of an inertial measurement unit (IMU) updated periodically with position fixes from a radiometric area correlation sensor. Other functions assumed are built-in-test (BIT), a digital fuze, and the flight control function.

The moderate complexity configuration chosen (Figure 4) utilizes a long range glide weapon aerodynamic configuration, represented by the GBU-15(V) with a planar wing module. This weapon is controlled during midcourse flight by navigation based on a combination of inertial sensor information (illustrated as an IMU module) and LORAN receiver. After navigation to the target area, the target would then be acquired by an electro-optical terminal seeker through use of a data link.

Additional functions of flight control, a digital fuze, and built-in-test are also included in this configuration.

The third configuration, selected to represent a lower level of complexity than the previous configurations, employs the GBU-15(V) with the cruciform wing module as the vehicle. This weapon, having a maximum range far beyond the lock-on capability of the selected Imaging Infrared Terminal Seeker, utilizes DME for midcourse guidance and a data link to allow target acquisition at long range. The only additional function assumed is the flight control function. (See Figure 5.)

For each of these three weapon configurations, the processing requirements for the subsystem will be determined for various interfaces within the subsystems. Then a point design for each configuration will be analyzed to determine the processing requirements for a fixed-design weapon. These point designs will form the basis for examining modularity requirements for the digital processor.

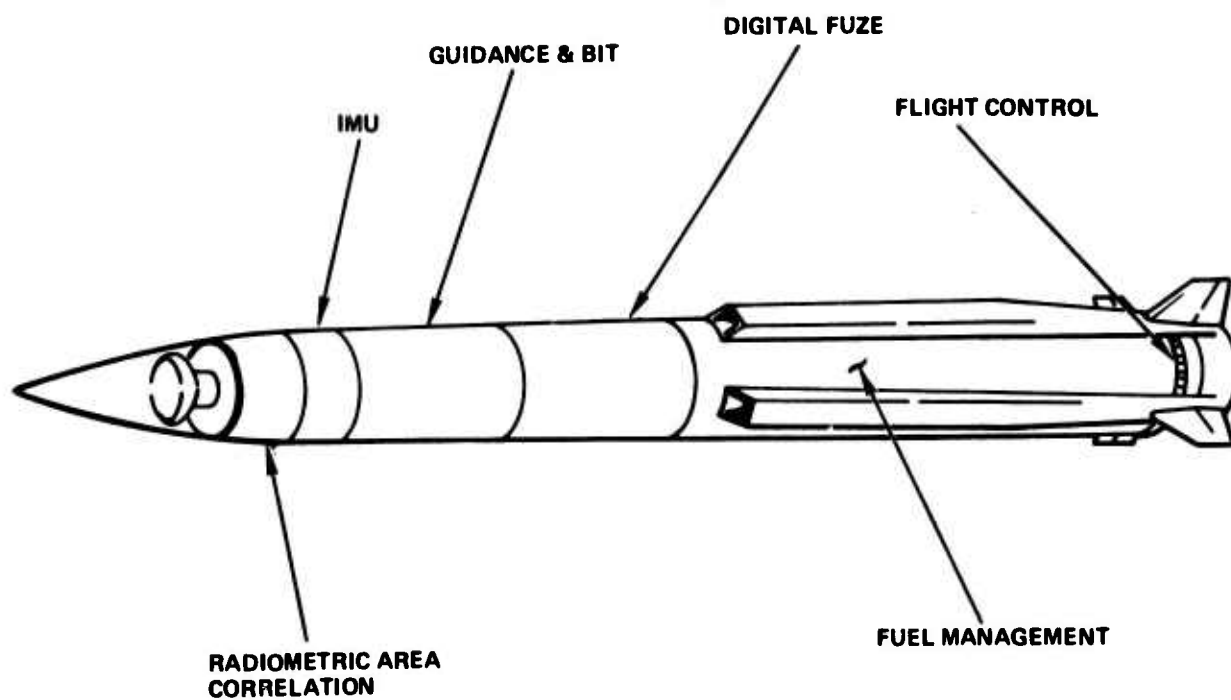


Figure 3. Candidate Weapon Configuration No. 1 (High Complexity)

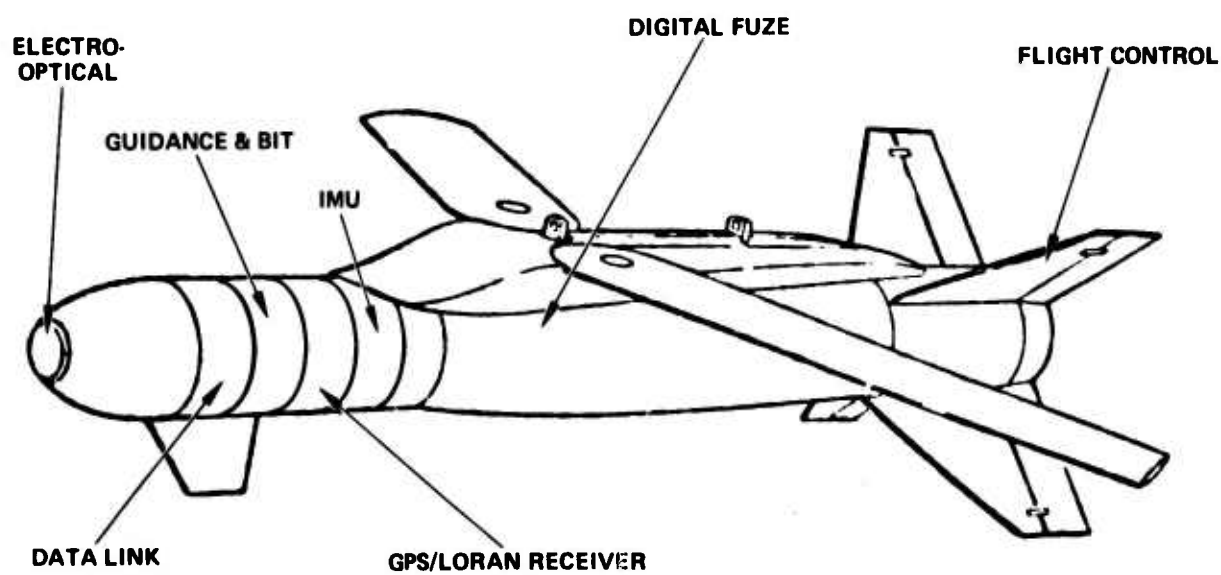


Figure 4. Candidate Weapon Configuration No. 2 (Moderate Complexity)

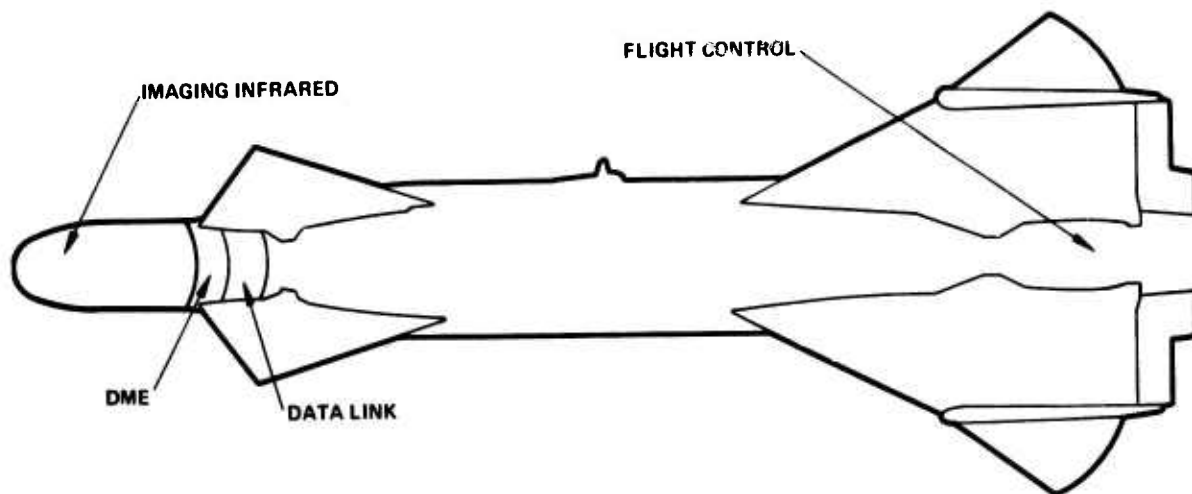


Figure 5. Candidate Weapon Configuration No. 3 (Low Complexity)

### SECTION III

#### DIGITAL SYSTEM DESIGN PROCEDURE

The design of a digital processing system to be a component of all possible configurations of a modular weapon system is a complex process. The most difficult problem is the determination of the role of the digital processor in each weapon configuration which will minimize the overall system cost. The problem is aggravated by probable addition of new subsystems (currently undefined) to the weapon system in the future resulting in a proliferation of weapon configurations. To maintain perspective on a problem which is potentially unbounded, the following procedure was adopted.

The system design process shown in Figure 6 was applied to each of the three weapon configurations determined in the previous section. Digital processor requirements were determined for various interface definitions within the subsystems of these configurations. These interface definitions ranged from treating the subsystem as a unit (minimum digital system requirement) to performing digitally all functions which are technically feasible. The results of these studies for the three configurations were correlated to determine a range of digital processing requirements within which to perform system optimization. The primary emphasis in the optimization process was placed on the partitioning of the system functions between the digital processor and dedicated processing elements. Various processor designs were performed to encompass the range of processing requirements as an integral part of this study.

To ensure that the results of this process were applicable to the total weapon system, other subsystems beyond those involved in the three configurations were studied in determining final digital processor requirements. These subsystems were examined in sufficient detail to validate the conclusions made on the basis of the three configurations. The remainder of this section is concerned with the detailed methodology used in the design process.

#### 3.1 SYSTEM DESIGN PROCESS

The system design process shown in Figure 6 delineates an orderly procedure which was followed in the design of the digital processor. The primary inputs to the process are the system level requirements which include not only performance, but also assembly and maintenance procedures and system operating philosophy. The system level requirements can be used to derive a set of functional requirements at the weapon configuration level, at the component subsystem level, and at the subsystem function level. The functional requirements are the basis for the partitioning study in which all functions of each weapon configuration are placed in one of four categories: analog, interface, special purpose digital, and general purpose digital. The final determination of the system partitioning

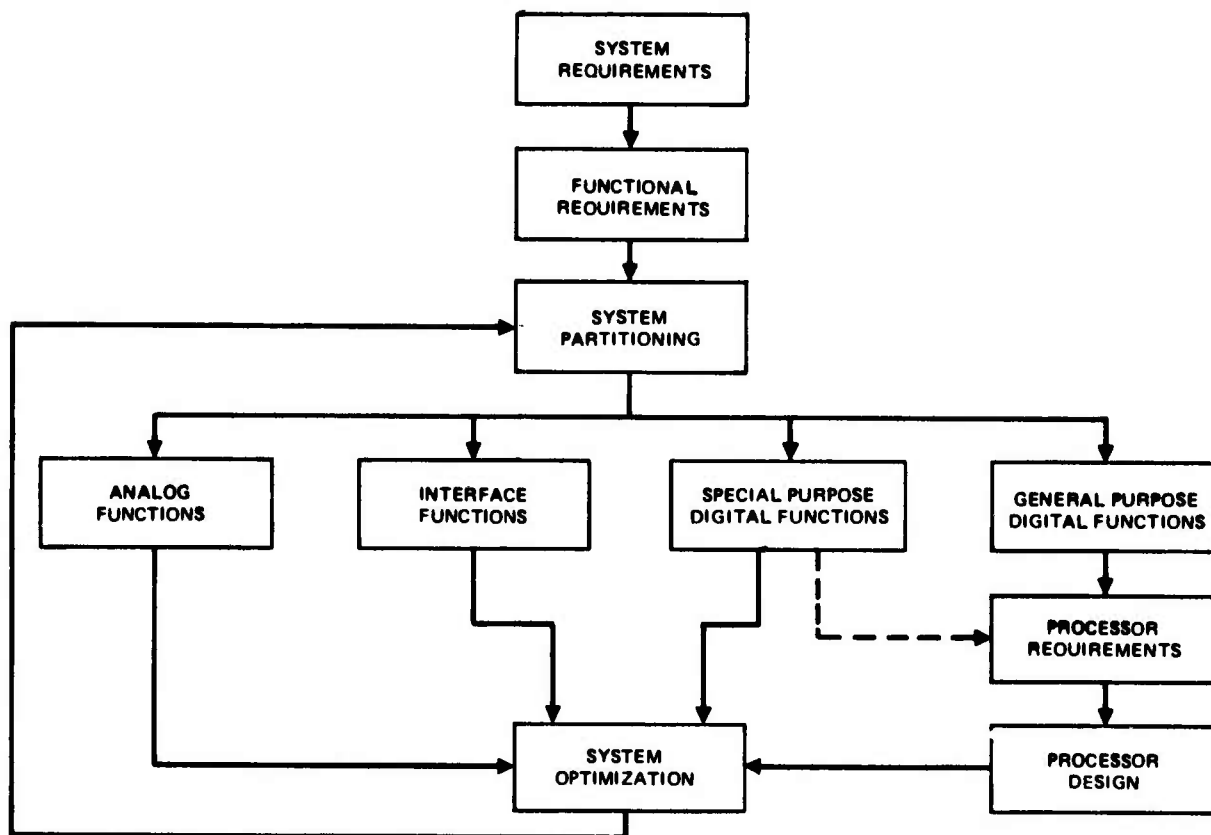


Figure 6. System Design Process

is made on the basis of the cost effectiveness of the resulting design. This system level optimization process must consider not only the effect of the partitioning on digital processor cost but also the effect on the costs associated with the other weapon subsystems. A detailed evaluation of subsystem costs for various functional partitionings was not performed in this study, but current implementation trends in similar subsystems were used to establish viable functional partitioning for this weapon system.

### 3.2 SUBSYSTEM CHARACTERIZATION

As a first step in the partitioning process, each subsystem was characterized as shown in Figure 7. The characteristics of interest are the data and control requirements (interface), the operations and processes performed by the subsystem, and the performance required of the subsystem as a component part of the weapon. The interface and performance requirements must be defined not only at the subsystem level but also for the internal functions of the subsystem. Partitioning of subsystem level performance requirements among the component functions of the subsystem is not inique but is generally implementation dependent. The key to the most cost-effective subsystem implementation is the use of the minimum cost technology which meets the requirements for each of the functions.

Many of the component subsystems either are in production or are in some stage of development. It was not the goal of this study to redesign these subsystems in digital technology. The reason for examining the internal functions of these subsystems was to establish requirements for typical functions which might be incorporated in the digital processor for advanced subsystems of the same type.

The selection of candidate digital processing functions from each subsystem was made on the basis of the operations and processes involved in the functions, the accuracy requirements, and the bandwidth of the interface signals. These functional characteristics were used in conjunction with previously determined digital processing requirements for functions with similar characteristics to perform a gross partitioning between digital and analog implementation.

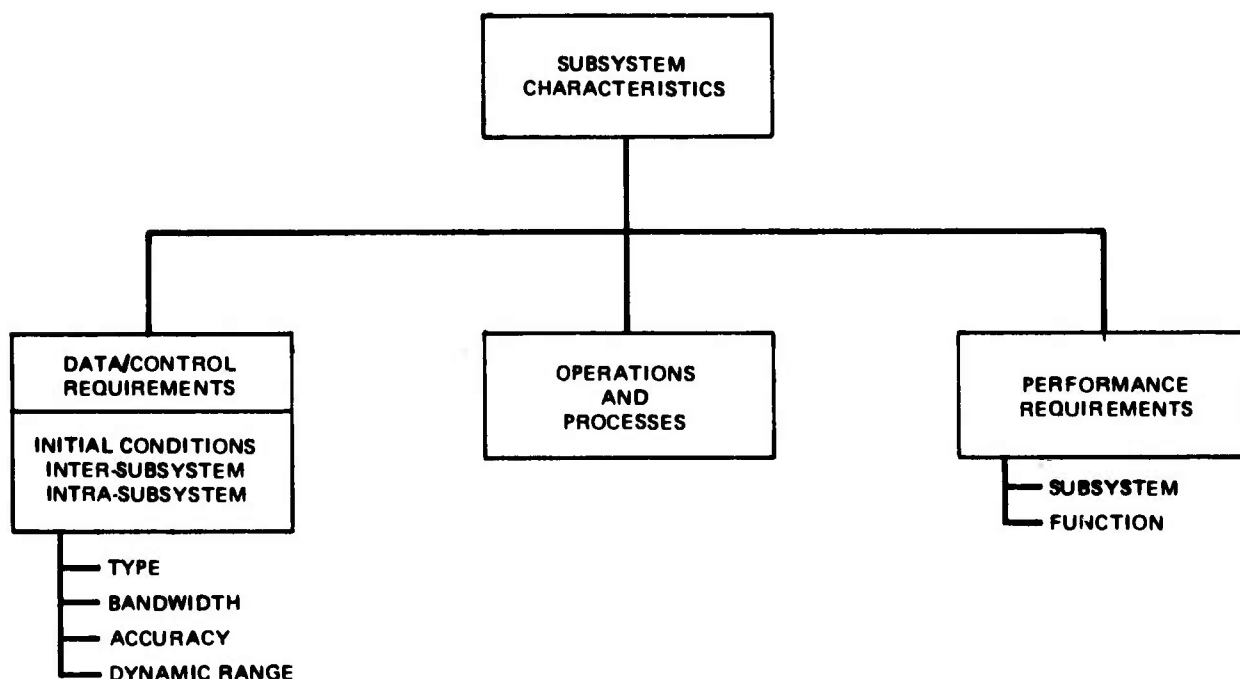


Figure 7. Subsystem Characterization

### 3.3 DIGITAL FUNCTION PARTITIONING

Digital processing requirements for each candidate subsystem function were determined by the procedure shown in Figure 8. The first step is the selection of an appropriate processing algorithm for the function. Since an algorithm is a method of performing the desired function within the functional requirements, the term may be applied to either analog or digital implementations. Every algorithm is an approximate solution to the ideal function, and the functional requirements define the allowable deviations from the ideal. The optimum algorithm considers the strong and weak points of the implementation technology to minimize implementation complexity. Consequently, the optimum algorithm is usually different

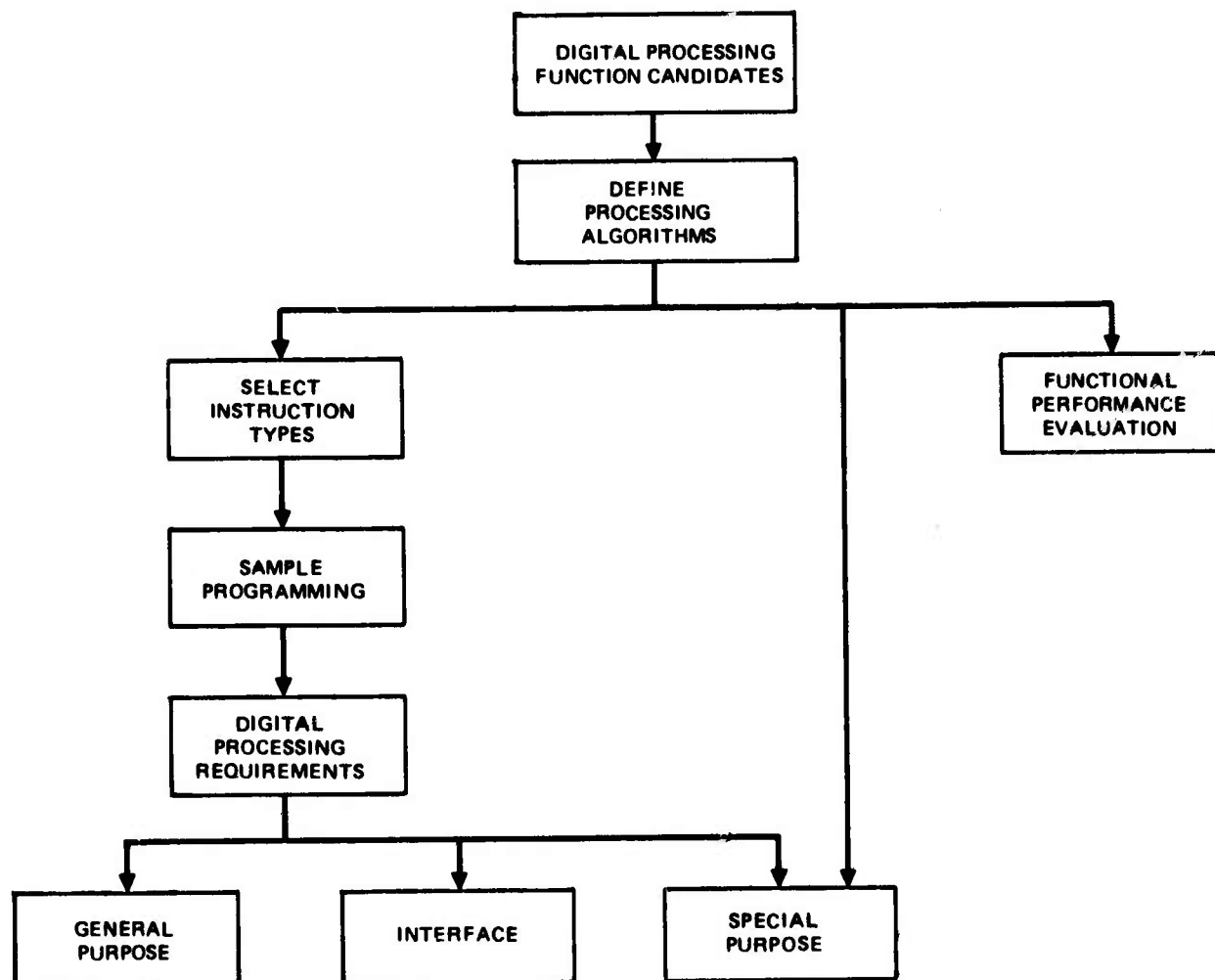


Figure 8. Digital Function Partitioning

for analog and digital implementations. Therefore, the procedure involved the identification of the ideal solution, when possible, and definition of digital processing algorithms for evaluation. For cases in which the ideal solution could not be determined, a digital approximation to the analog implementation was made.

The digital processing algorithms were evaluated from a performance viewpoint and processing requirements were determined. The generation of algorithms with improved performance was not a goal, but potential performance improvements were identified as a weighting factor for growth provisions. Digital processing requirements for each algorithm were evaluated by performing sample programming using appropriate instruction types. The instruction types were selected on the basis of the operations implied by the algorithms as shown in Table 1. The results of this evaluation were memory size (data and program), a sample instruction set, and instruction throughput requirements for the digital processor. A preliminary examination of these results was used to partition the digital functions into



TABLE 1. OPERATIONS AND INSTRUCTION TYPES

OPERATIONS	INSTRUCTION TYPE	ADDRESSING MODES
EXECUTIVE AND CONTROL DATA DISTRIBUTION INTERRUPT SERVICING  FILTERING INTEGRATION SCALING TRANSCENDENTAL FUNCTIONS LIMITING DECISION MAKING  ARRAY DATA PROCESSING	TRANSFER OF CONTROL ● UNCONDITIONAL ● CONDITIONAL	● INDIRECT ● DIRECT
	DATA MANIPULATION ● TRANSFER ● ARITHMETIC ● LOGIC ● INPUT/OUTPUT ● TEST	● REGISTER ● IMMEDIATE ● DIRECT ● INDEXED
	MACHINE CONTROL ● INTERRUPT CONTROL	

three categories: general purpose (software control), special purpose, and interface functions. A more definitive partitioning was made by determining the implications of various functional partitioning on digital processor design.

The processing requirements for the individual subsystem functions were now combined to determine total processing requirements for various weapon configurations. As a first step in this process, the functions were divided into two categories: weapon configuration common and configuration dependent. The functions were also screened for completeness, i.e., some weapon configuration functions are not attributable to any specific weapon subsystem but logically should be performed by the digital processor. The result of these analyses was a range of processor requirements for use in processor design tradeoffs.

### 3.4 DIGITAL PROCESSOR DESIGN

The digital processor design process is shown in Figure 9. Various digital processor architectures were examined to determine their ability to meet the range of processing requirements. This investigation included both digital processing system architectures (central versus distributed) and processor architectures (single CPU versus multi-processor). The applicable architectures and available digital component technology (in various semi-conductor families) were prime inputs to the processor implementation study. The system level requirements on weapon assembly and operating procedures were also used in determining viable processor implementations. Those processor implementations which were capable of performing over a relatively wide range of requirements were evaluated to determine cost versus performance factors which were used in system optimization studies.



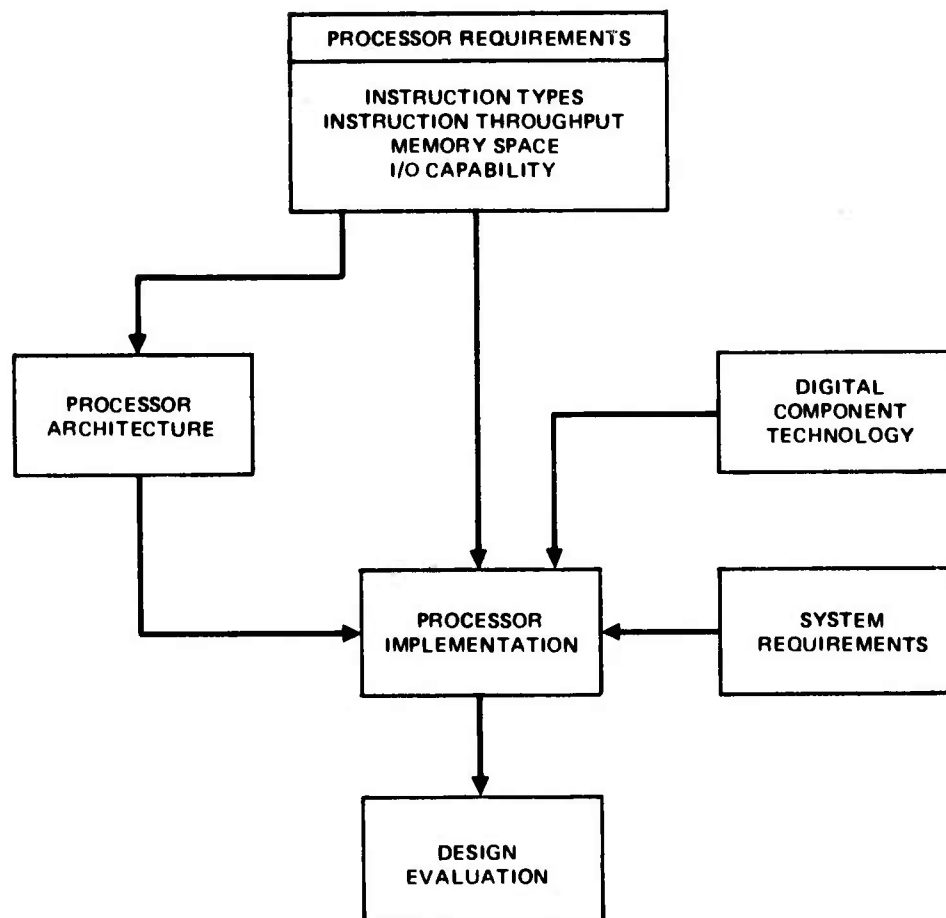


Figure 9. Digital Processor Design

The end result of the system design process is a specification of digital processing system requirements. These requirements are determined by applying a growth factor to the minimum capability required for weapon system integration and configuration common functions. This growth factor is necessary to accommodate system level growth in the integration and common functions since the system is not totally defined. To some extent, the processor growth capability may be utilized to perform some configuration dependent functions within the existing system definition. However, as a ground rule, a subsystem function should not be performed by the digital processor unless sufficient capability is available to perform the function for all weapon configurations.

This section has provided an overview of the analysis procedures and tradeoff criteria used in the design of the digital processor system. Section IV summarizes the analysis of subsystem requirements, and Section V summarizes the digital processor system tradeoffs.

## SECTION IV

### SUBSYSTEM DESCRIPTIONS AND REQUIREMENTS

The component subsystems of the weapon system have been examined to determine the applicability of digital processing to the functions of each subsystem. The subsystems have been collected according to their role in the weapon system, e.g., midcourse guidance. The characteristics of each subsystem have been analyzed, and candidate digital processing functions have been selected for requirements analysis. The effects of different interface definitions and processing algorithms on subsystem digital processing requirements are presented for each subsystem. Detailed functional requirements were not available for some subsystems of this weapon system. For these cases, requirements were generated using similar subsystems which are identified in the corresponding section.

The digital processing requirements for the functions of each subsystem include memory (program and operand), instruction throughput, and input/output data rate. The iteration rate of each algorithm is indicated where applicable. Instruction throughput requirements are shown for both short and long instruction types depending on instruction execution time. Long instructions consist of multiplication and division operations, and all other instruction types are in the short category.

#### 4.1 MIDCOURSE GUIDANCE SUBSYSTEMS

All midcourse guidance subsystems pertinent to this weapon system can be characterized as shown in Figure 10. Initial conditions on target position and weapon position and velocity are supplied by the avionics prior to weapon launch. The midcourse guidance subsystem updates weapon position and velocity data during flight based on environmental measurements. These measurements may be only the weapon environment (attitude, accelerations) or some combination of weapon and external environments. The guidance law operates on weapon position and velocity relative to the target position (or an intermediate aim point) to control the weapon trajectory. Although all of the midcourse guidance subsystems provide an essentially equivalent system function, there is considerable variation in their internal functions, as noted, which also results in variations in format, type, and quantity of initial condition data which must be supplied. Some of the midcourse guidance subsystems are also capable of providing terminal guidance information, and the variation in functional requirements is shown, where applicable.

##### 4.1.1 Strapdown Inertial Reference Subsystem

A functional block diagram of a strapdown inertial reference subsystem is shown in Figure 11. Initial conditions on weapon attitude, velocity, and position in an inertial coordinate frame are supplied by the avionics prior to weapon launch. After initialization, the weapon attitude in the inertial frame is updated on the basis of three-axis angular rate (or, alternatively, angle increment) data from body-fixed inertial sensors. Weapon velocity and position in the inertial frame

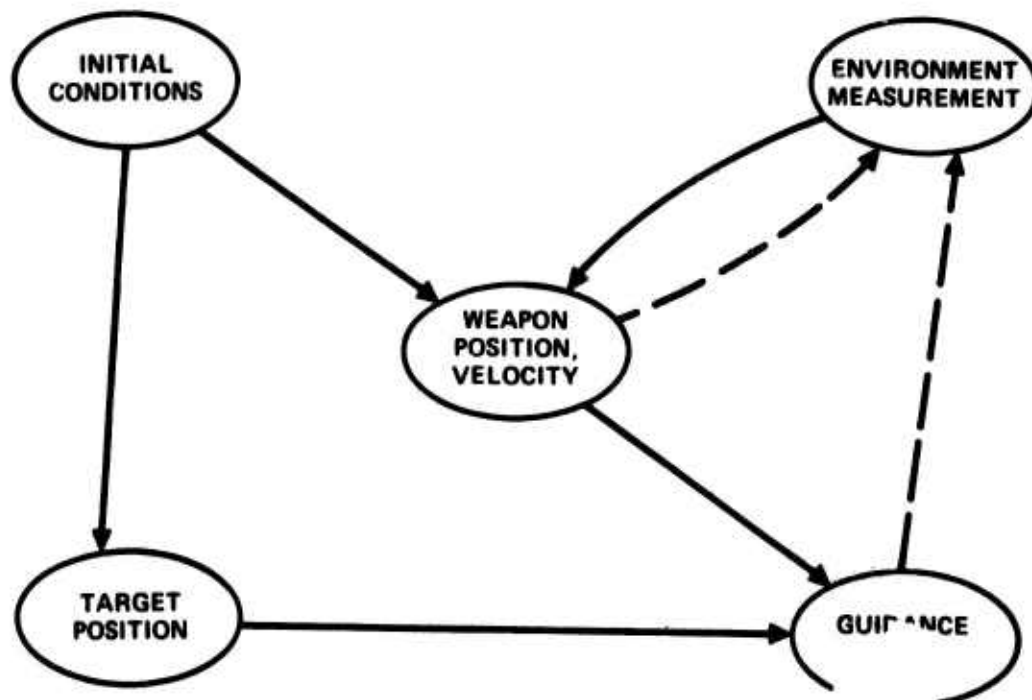
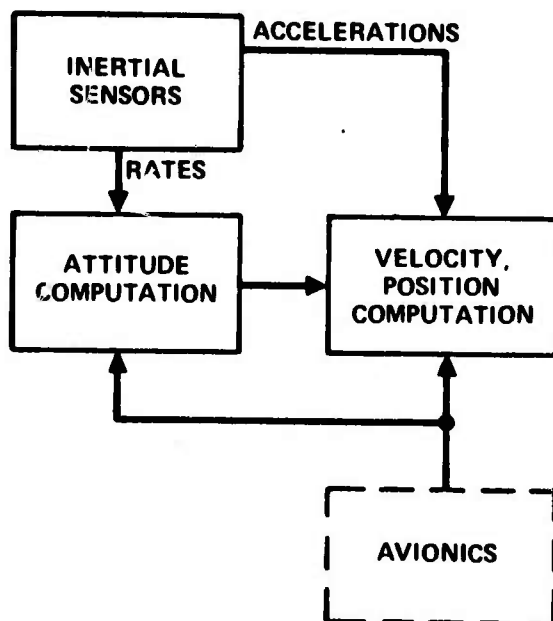


Figure 10. Midcourse Guidance Characterization.



INPUTS: INITIAL CONDITIONS – ATTITUDE (3), VELOCITY (3), POSITION (3)  
 RATES (3)  
 ACCELERATIONS (3)

OUTPUTS: VELOCITY (3), POSITION (3)

Figure 11. Strapdown Inertial Reference Block Diagram

are computed by transforming sensed acceleration (or velocity increment) data from three-axis body-fixed sensors into the inertial frame and performing the appropriate integrations. The candidate digital processing functions for this subsystem are the attitude, velocity, and position computation elements.

As a first step in determining the digital processing requirements for these functions, the subsystem performance requirements must be determined. The following assumptions were made on subsystem requirements:

(1) The strapdown inertial reference subsystem is not required to operate autonomously during the midcourse phase as a source of weapon inertial data but will always receive inflight corrections from another weapon subsystem. Formatting of correction data will be performed by the software.

(2) Autonomous subsystem performance requirements are most stringent when operating in conjunction with the radiometric area correlation (RAC) subsystem.

(3) The digital processing algorithms must provide accuracy compatible with the highest quality inertial sensors anticipated for tactical missile usage. Software compensation of low quality instrument errors is required.

(4) Mechanical alignment of the sensors is not compatible with subsystem accuracy requirements, and the misalignment must be corrected within the digital processor.

(5) Inertial reference attitude, velocity, and position data will be combined with target and aimpoint data using a generalized guidance law for appropriate weapon trajectory control.

(6) An altimeter is required for vertical channel stabilization if the companion subsystem does not provide altitude corrections.

Using these assumptions, the digital processing functions associated with the strapdown inertial reference are defined in Figure 12. The conversion of angle rates and accelerations to incremental angles and velocities may be performed either in the digital processor or the inertial sensors depending on the sensor output data format. The sensor compensation software corrects the sensor data using a combination of pre-stored data and data derived by the alignment function.

The attitude computation operates on the incremental angle data using a third order quaternion algorithm similar to the algorithm used in the ATIGS, a high quality inertial subsystem undergoing development by the Navy. The quaternions are used to generate a transformation matrix from body coordinates (sensor frame) to a locally level navigation coordinate frame with weapon position determined in latitude and longitude. The incremental velocity data is transformed to the navigation frame and summed to determine velocity. The velocity data is integrated to determine weapon position. Corrections are made for gravity and coriolis effects.

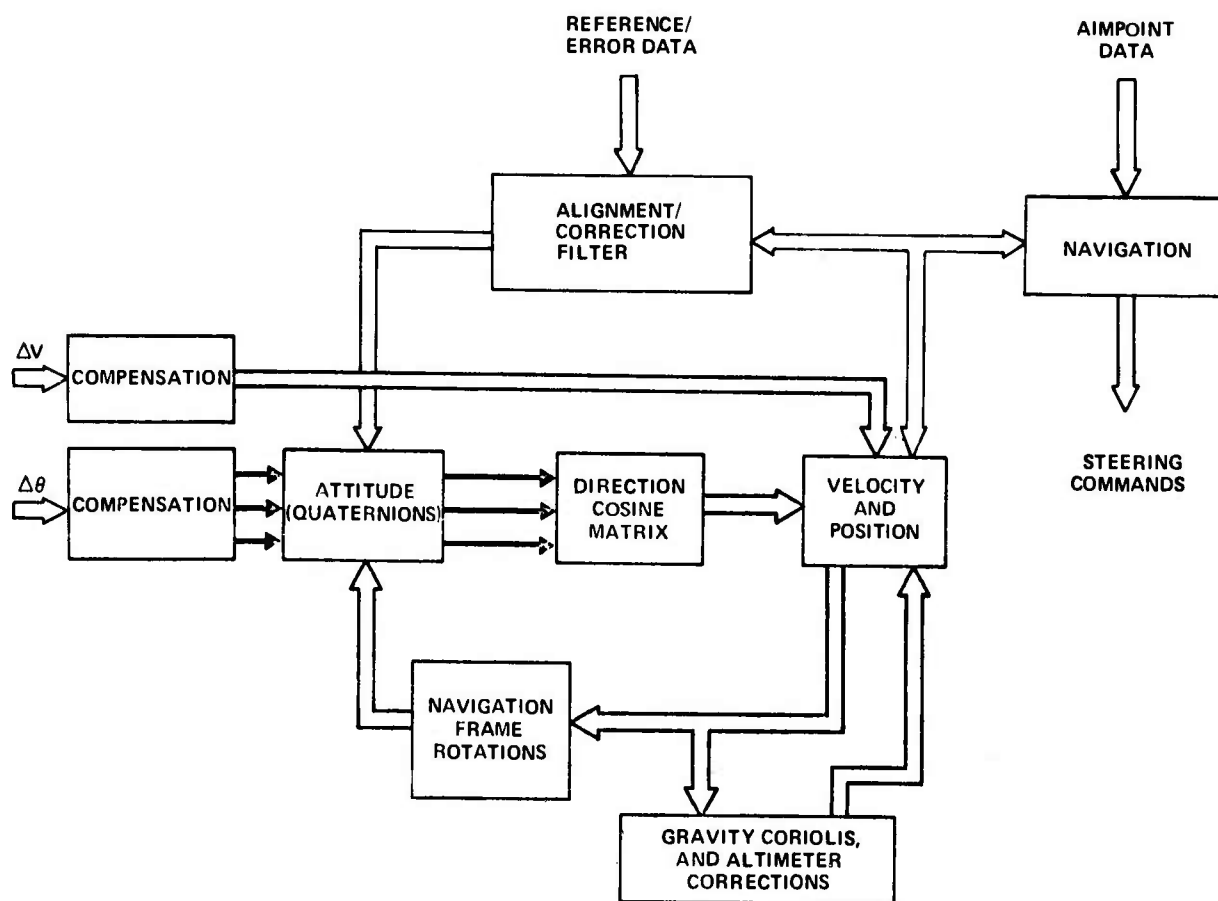


Figure 12. Inertial Reference Digital Processing Functions

The alignment filter uses velocity and position data from the avionics and the inertial reference to estimate attitude misalignment angles and gyro biases. A seven state Kalman filter is used for this function which is performed prior to weapon launch. To perform in-flight corrections, the data from the companion subsystem is formatted to serve as reference input data (or measured error data, depending on subsystem) for the Kalman filter. Corrections are made to the inertial reference data (attitude, velocity, position) on the basis of the bandwidth of data supplied by the companion subsystem. For example, the RAC subsystem data will be used to correct only position, while LORAN or GPS data has sufficient bandwidth to allow estimation of errors in all parameters.

The navigation function generates steering commands for the flight control based on line of sight (LOS) angle and LOS angle rate data derived from weapon velocity and position relative to a desired aimpoint and approach angle for appropriate operation of the companion subsystem. These parameters are discussed in more detail in the descriptions of the other midcourse subsystems.

The digital processing requirements for the strapdown inertial reference functions are shown in Table 2. These data assume the highest expected iteration rate for the configuration dependent functions

TABLE 2. PROCESSING REQUIREMENTS - STRAPDOWN  
INERTIAL REFERENCE

FUNCTION	PROGRAM SIZE	OPERAND MEMORY	THROUGHPUT		COMMUNICATION REQUIRED
			SHORT	LONG	
IMU COMPENSATION, ATTITUDE, VELOCITY, POSITION CALCULATION	800 (988)	150 (310)	78.5 KOPS *	10.7 KOPS *	6 AT 100 Hz 1 AT 10 Hz
ALIGNMENT/ CORRECTION FILTER	(550)	(190)	10 KOPS **	1.35 KOPS **	4 AT 1 Hz
NAVIGATION	150	40	2 KOPS	0.2 KOPS	2 AT 10 Hz
* MEASURED COMPUTATION TIME - 1 MILLISECOND/ITERATION OF 100 Hz COMPUTATIONS (DP1 BREADBOARD)					
** MEASURED COMPUTATION TIME - 8 MILLISECOND/ITERATION (DP1 BREADBOARD)					

(error formatting and in-flight Kalman filter). Some of these functions have been implemented in the software for the DPI breadboard system and the measured parameters are indicated in parentheses, where available.

#### 4.1.2 Radiometric Area Correlation Subsystem

The functions of the RAC subsystem are shown in the block diagram of Figure 13. This subsystem operates in conjunction with an inertial reference (IR), for weapon guidance and may be used in both midcourse and terminal flight phases. The RAC subsystem correlates sensed maps with pre-stored reference maps to determine inertial reference position error data. The coordinates and orientation of each reference map are used as aimpoint data in the strapdown inertial reference for weapon trajectory control.

The weapon position and attitude data from the inertial reference and reference map position data are used in the scan control function to dynamically control the position of a gimballed radiometer. The radiometer utilizes a resonant scan in the azimuth plane, and the scan control function controls the pitch plane position of the scan and isolates the scan from weapon motion. The scan control function also provides sampling pulses to the sensed map generation function.

The output of the radiometer is continuously input to the sensed map generation function. The sampling pulses cause the radiometer output to be sampled and stored at intervals corresponding to the resolution cell size of the reference map. The size of each sensed map varies during weapon flight. The cell size also varies during flight. Since the radiometer antenna aperture is fixed, the weapon altitude must be controlled to ensure that the available resolution is

commensurate with cell size requirements. After each sensed map is stored, the correlator function determines the best match (highest correlation) of the sensed map data with all available positions of the sensed map within the reference map. The position of the best match relative to the center of the reference map corresponds to the IR position error normalized to the cell size.

The candidate digital processing functions are the scan control and the correlator, both of which are performed digitally in the developmental model of this subsystem. The scan control function consists of two coordinate transformations which determine the position of the desired sensed map sample points in antenna coordinates. The sample point coordinates in an inertial frame are first transformed to the weapon body coordinate frame using equations defined in Lockheed report TR-925. This transformation uses a direction cosine matrix which is available in the strapdown inertial reference subsystem. The transformation from body to antenna coordinates is then performed on the basis of measured antenna gimbal angles. The sampling pulses are generated with 100-microsecond time resolution in the second transformation when the antenna position matches the desired coordinates. The digital processing requirements associated with the scan control functions are shown in Table 3.

Two different processing algorithms were investigated for the correlator function. These algorithms were the normalized deviation product

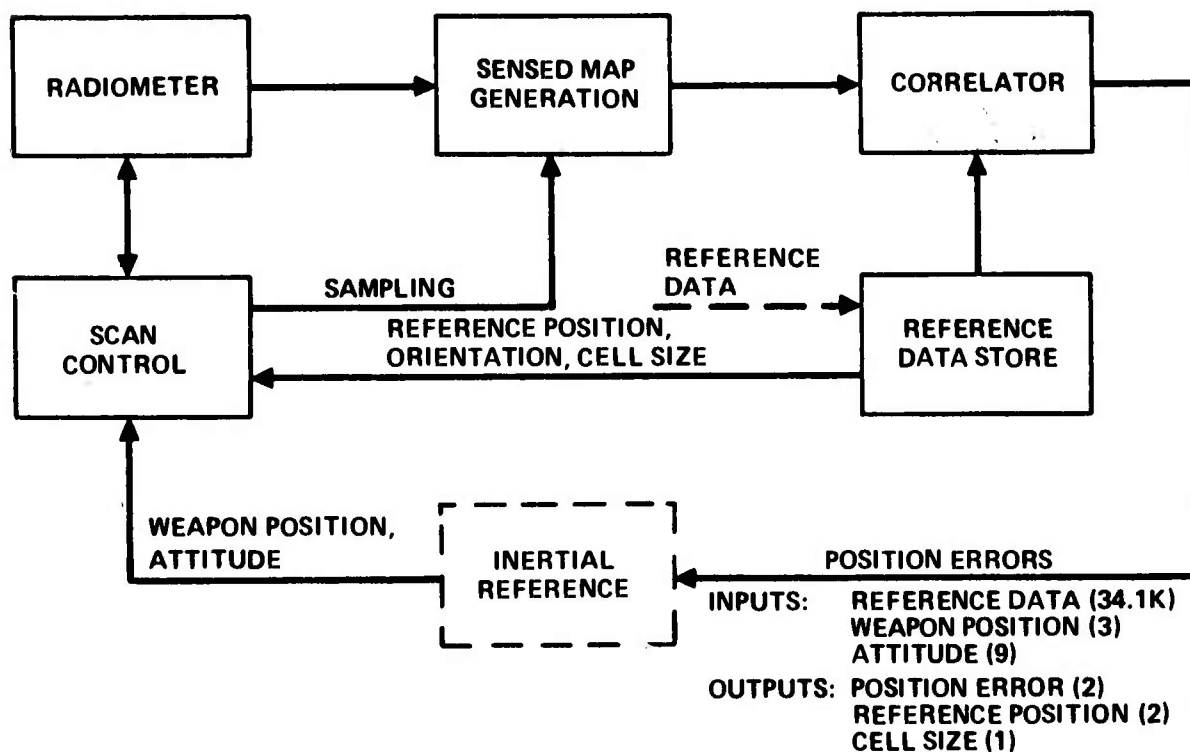


Figure 13. Radiometric Area Correlator Block Diagram



TABLE 3. PROCESSING REQUIREMENTS - RAC SUBSYSTEM

FUNCTION		PROGRAM SIZE	OPERAND MEMORY	THROUGHPUT		COMMUNICATION REQUIRED
				SHORT	LONG	
SCAN CONTROL: INERTIAL-MISSILE COORDINATE CONVERSION		250	85	73 KOPS	8 KOPS	40 AT 100 Hz 1 PULSE AT 14 Hz 6 AT 0.25 Hz
SCAN CONTROL INERTIAL-ANTENNA COORDINATE CONVERSION		300	100	455 KOPS	138 KOPS	7 AT 10 kHz 12 AT 100 Hz 4 AT 0.25 Hz 1 PULSE AT 10 kHz
CORRELATOR NPDM ALGORITHM	I/F1	175	9.5-35K	15.3 MOPS	2.79 MOPS	1 AT 448 Hz
	I/F2	100	30	434 KOPS	98 KOPS	3 AT 10.9 kHz 2 AT 1 Hz
	I/F3	60	15	217 KOPS	0	1 AT 10.9 kHz
CORRELATOR SSDA ALGORITHM		100	8.5-34K	9.94 MOPS	0	1 AT 448 Hz

moment (NPDM) used in the RAC developmental model and the sequential similarity detection algorithm. The operations required for these algorithms to determine each correlation value (one sensed map position within the reference array) are shown in Figure 14. The remainder of the correlator function requires the determination of the best correlation point (maximum value for NPDM, minimum value for SSDA) using the correlation values calculated. The SSDA potentially allows early truncation of the calculation point values, since the accumulation may be stopped when it exceeds the previous minimum value (or some preset threshold). However, this advantage was not assumed in the determination of processing requirements for the correlator function.

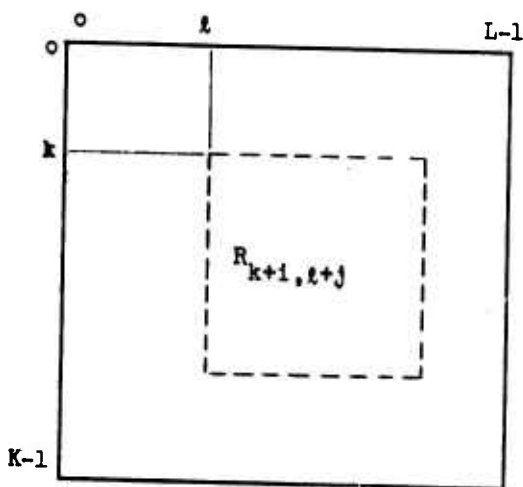
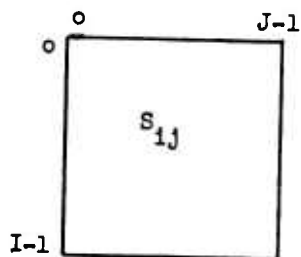
Digital processing requirements for the two algorithms (and alternative interface definitions in the NPDM algorithm) are shown in Table 3. These numbers are based on a requirement to calculate 1089 correlation points for a 32 x 32 sensed array within 0.8 second.

#### 4.1.3 LORAN Subsystem

The LORAN subsystem and associated functions are shown in Figure 15. The designation of the master and slave stations in the LORAN network appropriate to the weapon mission is performed during the weapon assembly process. The receiver processes the signals from the designated stations and outputs a set of time difference data corresponding to the differences in time-of-arrival between the signals of the master station and each of the slave stations. The time difference data are then processed to determine the receiver position.



SENSED ARRAY



REFERENCE ARRAY

Interface 1

$$RSAS(k, l) = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} R_{k+i, l+j}$$

$$R2S = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} R_{k+i, l+j}^2$$

$$RSS = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} R_{k+i, l+j} s_{ij}$$

Interface 2

$$SAS = \sum_{m=0}^{I-1} \sum_{n=0}^{J-1} s_{mn}$$

$$S2S = \sum_{m=0}^{I-1} \sum_{n=0}^{J-1} s_{mn}^2$$

$$\|s\|^2 = S2S - SAS^2/IJ$$

$$\|R_{kl}\|^2 = R2S - RSAS(k, l)^2/IJ$$

$$R_{kl, S} = RSS - [SAS][RSAS(k, l)]/IJ$$

$$C(k, l) = [R_{kl, S}]^2 / \|R_{kl}\|^2 \|s\|^2$$

Interface 3

Find location of maximum value of  $C(k, l)$

NPDM ALGORITHM DEFINITION

$$RAV(k, l) = \frac{1}{IJ} \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} R_{k+i, l+j}$$

$$SAV = \frac{1}{IJ} \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} S_{i,j}$$

$$d(k, l) = \frac{1}{IJ} \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} \left| R_{k+i, l+j} - S_{i,j} - RAV(k, l) + SAV \right|$$

Determine location of minimum value of  $d(k, l)$

#### SSDA ALGORITHM DEFINITION

Figure 14. Correlation Algorithms

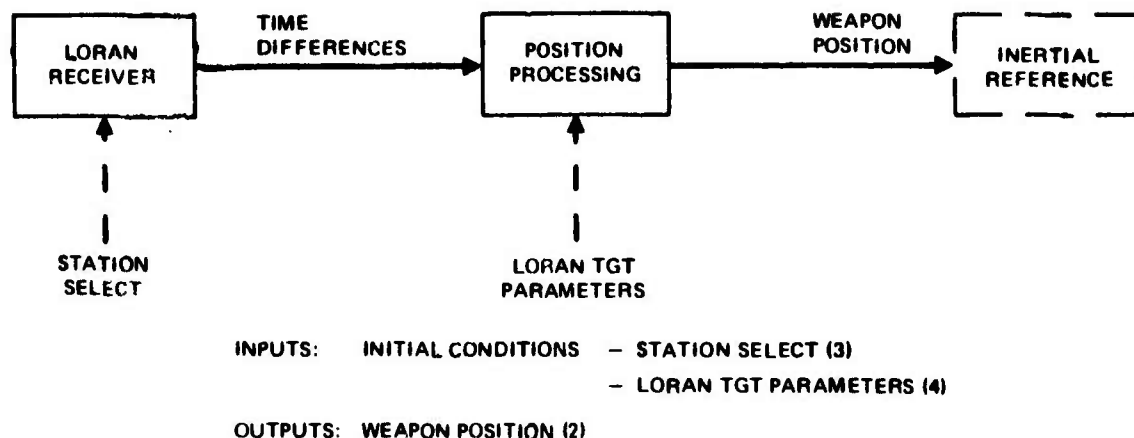


Figure 15. LORAN Subsystem Block Diagram

Only the position processing is a candidate digital function because the digital portions of the receiver functions are included in the receiver packages available from many different vendors. Several different algorithms are available to determine the weapon position from the time difference data. These algorithms involve different initialization parameters for the LORAN network and are mission dependent. The algorithm used in the study operates on the time difference data to derive weapon position relative to the target position in the inertial reference navigation frame. This is accomplished by first subtracting the measured time difference for each slave station from the time difference for that slave at the target position. The resultant differences for a pair of slave stations are transformed to latitude and longitude differences using gradient data corresponding to the target position in the LORAN network. The gradient data were assumed to be input either during weapon assembly or by the avionics prior to weapon launch. The relative position data are output to the inertial reference subsystem to correct IR errors as discussed in 4.1.1. The processing requirements for this function are shown in Table 4.

#### 4.1.4 E-O Data Link Subsystem

The function of the E-O data link subsystem are shown in Figure 16. The characteristics used in the study are based on the data link subsystem of the GBU-15 weapon system. This subsystem provides in-flight control of both the flight control subsystem and either an E-O or IIR terminal guidance sensor by means of operator action. The operator actions are based on his monitoring the scene within the view of the terminal sensor. This is accomplished using the transmitter portion of the data link to send the terminal sensor video signal to the launch aircraft. The operator controls the weapon mode by means of command messages which are processed in the receiving portion of the data link.

Because of the data bandwidths involved, only the message decoding function is a candidate for digital processing. This function requires

that the bits in the command message be decoded to determine its component data and control signals, and the formatting and distribution of the signals to the appropriate subsystems. The requirements for this function are given in Table 5.

TABLE 4. PROCESSING REQUIREMENTS - LORAN SUBSYSTEM

PROGRAM SIZE:	100
OPERAND MEMORY:	50
COMMUNICATION REQUIRED:	5 AT 1 Hz
THROUGHPUT:	100 OPS (SHORT) 10 OPS (LONG)

TABLE 5. PROCESSING REQUIREMENTS - EO DATA LINK

PROGRAM SIZE:	100
OPERAND MEMORY:	20
COMMUNICATION REQUIRED:	(4 + 11 LOGIC) AT 30 Hz
THROUGHPUT:	17 KOPS (SHORT)

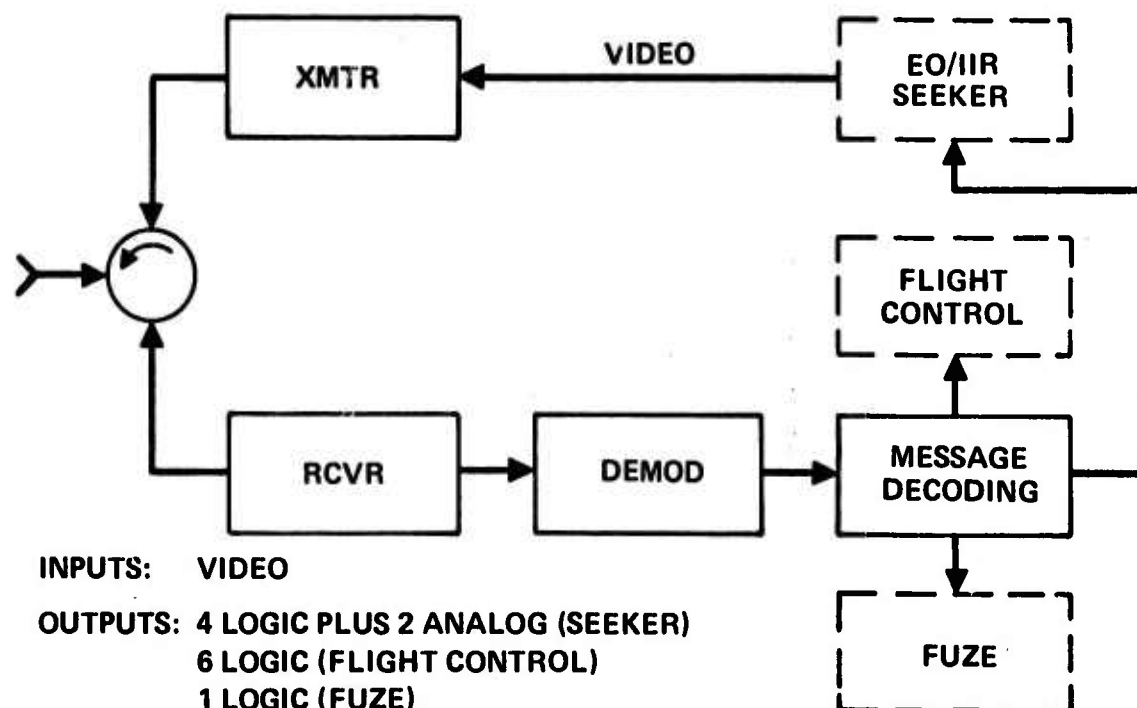


Figure 16. Electro-Optical Data Link Subsystem Block Diagram

#### 4.1.5 TERCOM Subsystem

The functions of the TERCOM (terrain contour matching) subsystem are shown in Figure 17. This subsystem matches measured terrain contours (strip map) with stored reference data to determine weapon position errors. The output of the radar terrain sensor (radar altimeter) is sampled when the weapon position data from the inertial reference match the position of the stored reference data. The reference altitude sensor (barometric altimeter) is also sampled to measure mean weapon altitude. The vertical accelerometer data provide isolation of vertical weapon motion during strip map generation. These data are combined to determine terrain elevation at each sample point. For this study, each strip map is assumed to be a sequence of data from 64 sample points. The strip map data are then filtered to remove both mean elevation and mean elevation rate. The resulting map data are then correlated with the stored reference data to find the best position match. The deviation of the match point relative to the center of the reference data is the position error normalized to the map cell size.

The candidate digital processing functions for this subsystem are the filtering of strip map data and the contour matching. The filtering function is performed by a 51-point transversal filter, requiring a total of 114 data samples for each strip map. The contour matching is performed by a one-dimensional implementation of the SSDA discussed in subsection 4.1.2. The allowable time to perform these functions was assumed to be 4 seconds in deriving the requirements shown in Table 6.

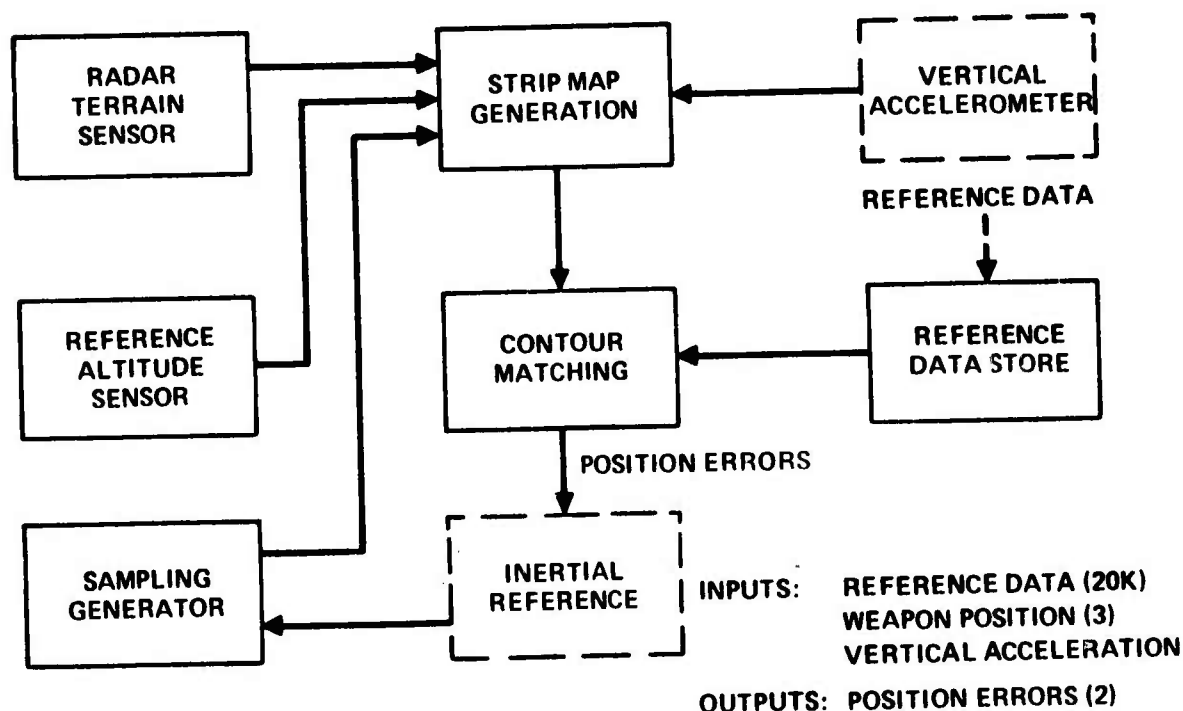


Figure 17. TERCOM Subsystem Block Diagram

TABLE 6. PROCESSING REQUIREMENTS - TERCOM SUBSYSTEM

FUNCTION	PROGRAM SIZE	OPERAND MEMORY	THROUGHPUT		COMMUNICATION REQUIRED
			SHORT	LONG	
FILTERING	50	100	1.1 KOPS	0.16 KOPS	1 AT 3 Hz 3 AT 100 Hz
POSITION ERROR	100	1755 + (10.5K BULK)	235 KOPS	8 OPS	1584 AT 1/300 Hz* 2 AT 0.25 Hz
*TRANSFER OF REFERENCE DATA FROM BULK STORAGE TO OPERAND MEMORY					

#### 4.1.6 DME Subsystem

The DME subsystem consists of a transmitter and command receiver which operate as a transponder, enabling aircraft or ground stations to determine weapon range data which are used to calculate weapon position in three coordinates. The computations relating weapon and target position are performed by one of the airborne or ground stations to generate weapon steering commands. These steering commands are transmitted to the weapon DME subsystem by means of an RF link, decoded and formatted for use by the flight control subsystem.

The only candidate digital processing function is the decoding and formatting of the command data. Insufficient information on the characteristics of this function was available to assess the associated processing requirements. However, the requirements for these functions in the E-O data link subsystem should be representative.

#### 4.1.7 Global Positioning System (GPS) Subsystem

The GPS guidance concept utilizes the transmissions from four earth-orbiting satellites to measure range to each satellite and thus determine missile position in three coordinates. The concept provides jam-resistant operation by relying heavily on the integration with an onboard inertial reference, coupled with pseudorandom coded satellite signals. The functional relationship of the GPS receiver, including the receiver process controller (RPC), with other missile subsystems is shown in Figure 18. (See Table 7.)

A more detailed illustration of the principal GPS guidance processing functions and their functional relationship is shown in Figure 19. GPS receivers which have been designed for aircraft, ship, and man-pack applications, provide digital measurements of pseudorange (prefixed "pseudo" because the measurements are relative signal time-of-arrival rather than absolute range) and delta range with the precise time of reception. These pseudorange measurements, after correction for known timing errors, are then utilized by the guidance (suboptimal) filter to compare with an internally derived estimate of satellite ranges.

TABLE 7. GPS GUIDANCE SUBSYSTEM DIGITAL PROCESSING REQUIREMENTS

THROUGHPUT: 300 KOPS	INSTRUCTION SET: GENERAL PURPOSE
COMPUTATION WORD LENGTH: 16 BITS	INTERRUPT CAPABILITY
DOUBLE PRECISION CAPABILITY	DIRECT MEMORY ACCESS
FUNCTION	MEMORY REQUIREMENT
SUBOPTIMAL FILTER	5200
RECEIVER DATA PROCESSING	6900
RECEIVER AIDING	
OUTPUT DATA PROCESSING	
INERTIAL NAVIGATION	4900
EXECUTIVE	3200
INTERRUPT HANDLING	
I/O ROUTINES	
INITIALIZATION	
SELF TEST	
SUBROUTINE LIBRARY	2200
TOTAL	22,400

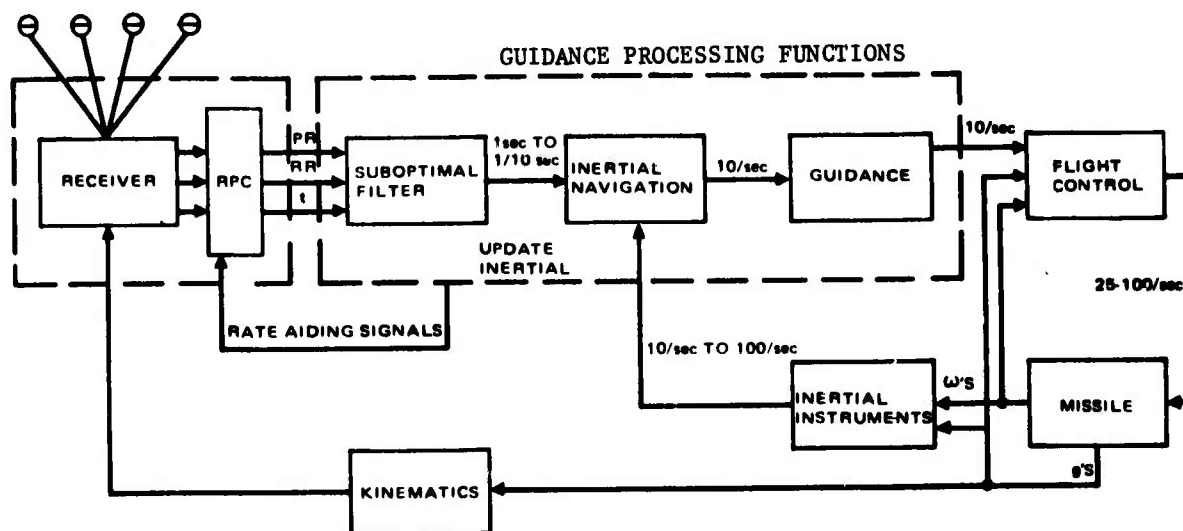


Figure 18. GPS Subsystem Functional Flow Diagram

The differences in range are then used by the filter to estimate current errors in position and velocity as well as the magnitude of the most probable causes of navigation error (such as misalignment). The precise measurements of signal reception time are required by the ephemeris calculations to accurately determine the position and velocity of each satellite at the time of transmission. The estimated inertial navigation errors provided by the guidance filter are then used to update the best estimates of missile position and velocity.



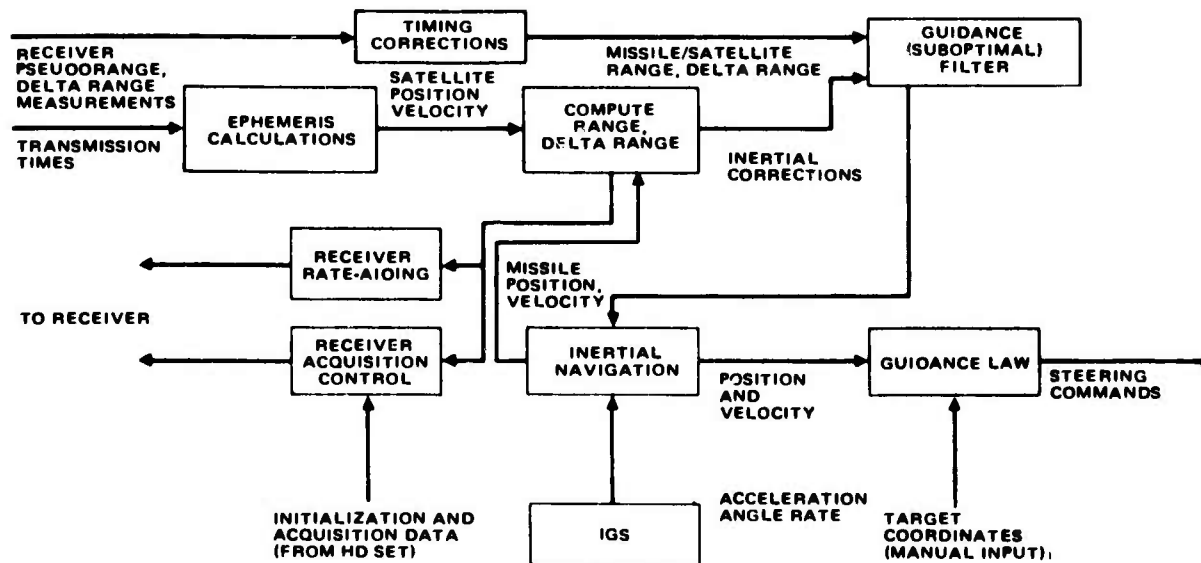


Figure 19. Principal GPS Guidance Processing Functions

The position and velocity estimates from the inertial navigation function are used by the guidance law to generate steering commands as input to the autopilot or flight control function. This same position and velocity information is also used to perform the receiver acquisition control and rate-aiding functions. The receiver acquisition control function must also accept the initialization and acquisition data from the aircraft GPS (HD) set and control initial receiver acquisition.

Estimates of the digital processing requirements were made by modifying the estimated processing requirements for the current SAMSO aircraft receiver demonstration program system, taking into account the differences in functional requirements between the aircraft navigation and missile guidance problems. Several factors have combined to cause the memory requirement estimates to be conservative. Firstly, the computer programs have been written in a higher-order language, resulting in inefficiency in generating the resultant machine language program. The developmental nature of the SAMSO demonstration system required additional programming for system test and instrumentation data collection. Also, no optimization of developmental software is warranted. Therefore, significant uncertainty remains regarding the actual requirements for a tactical weapon. Further development of actual missile processing algorithms and corresponding software must be undertaken to create a more accurate requirements estimate.

#### 4.2 TERMINAL GUIDANCE SUBSYSTEMS

The generic characterization of the terminal guidance subsystems of this weapon system is shown in Figure 20. Most terminal guidance techniques utilize some type of onboard target sensor for improved guidance accuracy. This, in turn, implies that the sensor must be pointed at the target, and that the target characteristics must be supplied to the guidance processing.



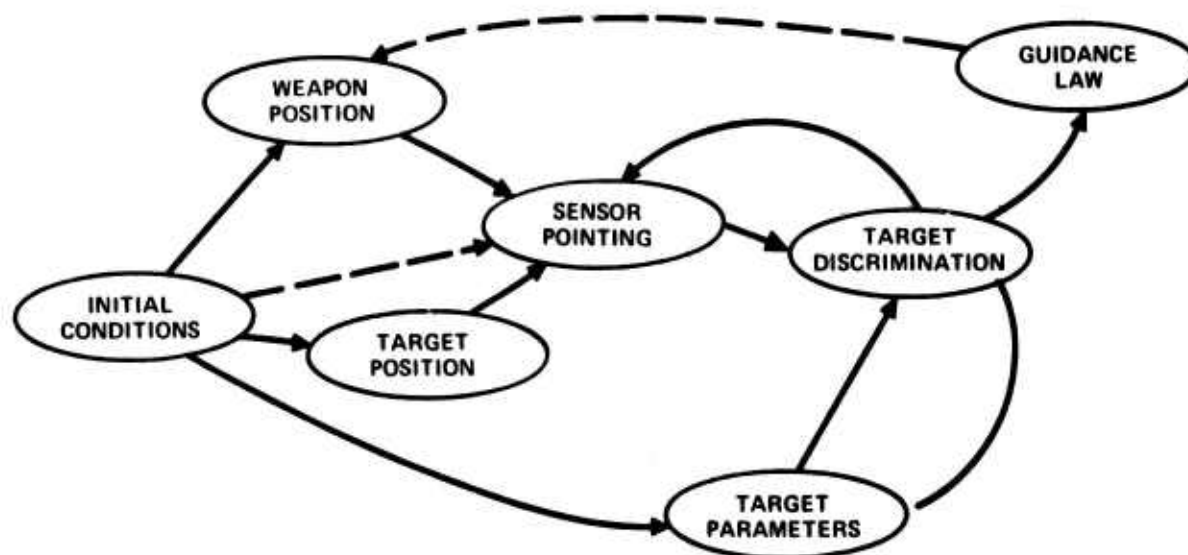


Figure 20. Terminal Guidance Characterization

Sensor pointing data can be derived by onboard computations on the basis of weapon and target position data from a midcourse guidance subsystem, e.g., inertial reference. Alternatively, sensor pointing data can be supplied by operator action, e.g., E-O seeker slew commands by means of the umbilical or E-O data link.

The data from the target sensor are processed to discriminate between target and background. This discrimination process is based on target parameters which may be preselected in the subsystem or inserted as initial conditions. In some subsystems, measured target parameters are used to update the initial conditions for improved discrimination.

Sensor pointing information, usually line of sight (LOS) angle or LOS rate, is processed through a guidance law to control weapon trajectory by means of the flight control subsystem.

#### 4.2.1 Electro-Optical and Imaging Infrared Seeker Subsystems

The functions of the E-O and IIR subsystems and associated subsystems are shown in Figure 21. The functions of these two seeker subsystems are essentially identical with the primary differences being in the implementations of the sensor and scan converter functions. The outputs of the scan converter are wideband video signals which are sent to the video processor and to the E-O data link subsystem (see subsection 4.1.4). The video processor amplifies the analog video signal to normalized level for input to the threshold function which discriminates between target and background. The threshold output is sent to the tracking function which derives signals for steering, sensor pointing, and video processor control.

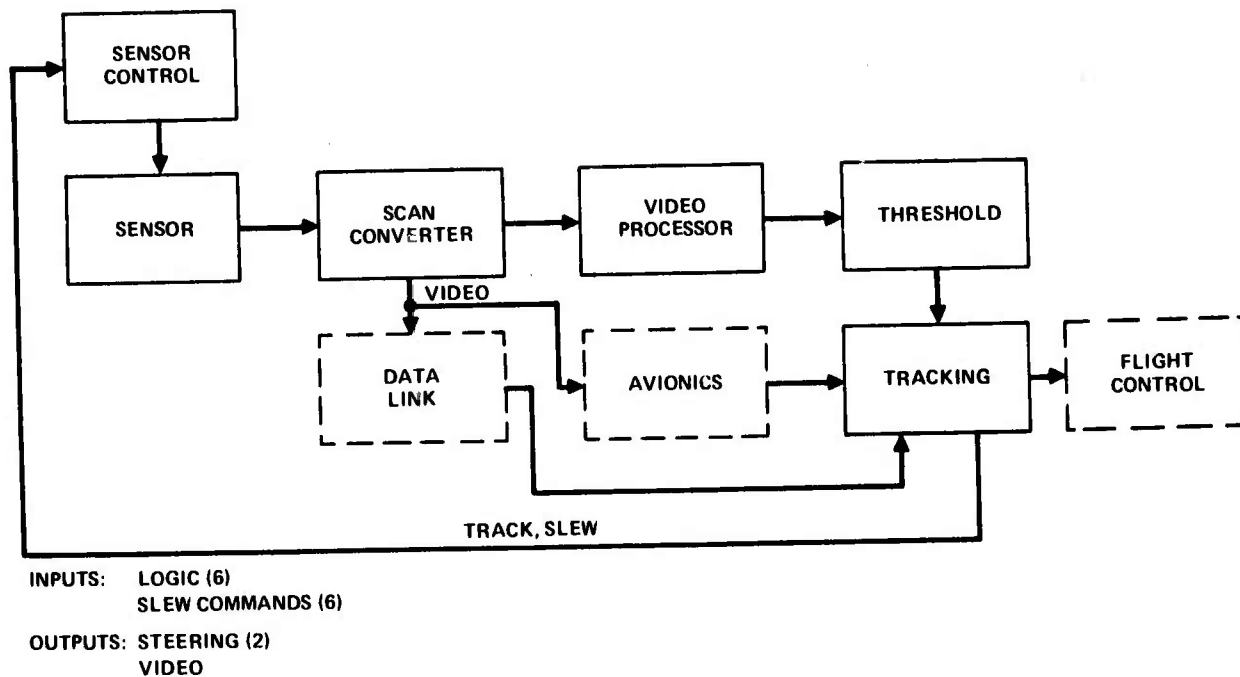


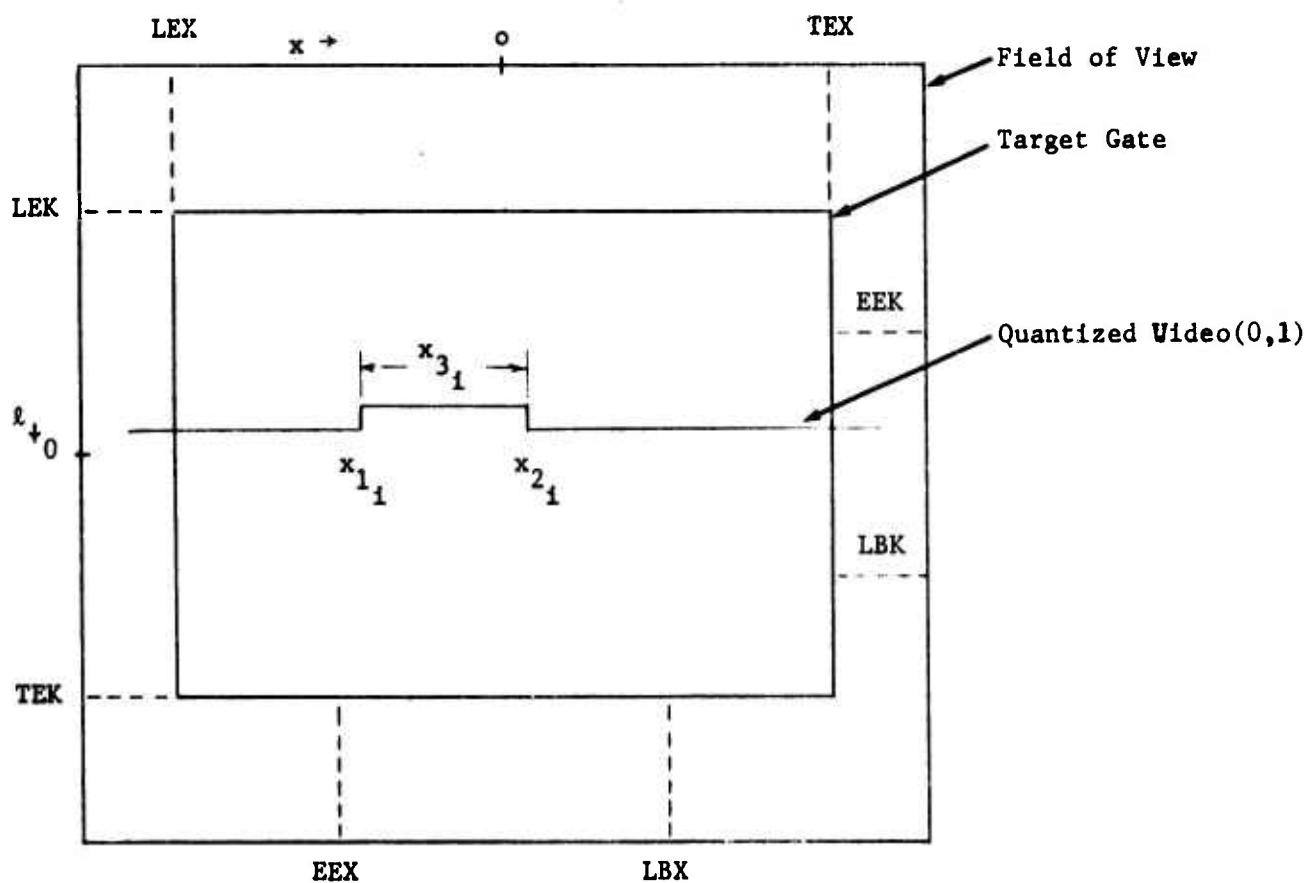
Figure 21. EO/IIR Seeker Block Diagram

The candidate digital processing functions for these subsystems are the control of the video processor, the thresholding, and the target tracking. Two algorithms which perform these functions were investigated in this study. One algorithm is used in the ATVS system under development by the Army. The second algorithm is the mathematically optimum for discrimination between target and a Gaussian background. The two algorithms are presented in Figures 22 and 23. The processing requirements associated with these two algorithms are shown in Figure 24 for various partitioning between special purpose processing and general purpose processing.

#### 4.2.2 Radiometric Contrast Seeker Subsystem

The functions of the radiometric contrast seeker subsystem are shown in Figure 25. Target line of sight data is used to point the gimbaled sensor to the designated target. For inflight acquisition, this information can be derived from the inertial reference weapon position data and the target position initial condition. The wideband output of the sensor is amplified in the receiver, and target discrimination is performed by detecting the derived sensor pointing error signals which control the sensor position. The pointing error data are output to the flight control subsystem as steering commands.

Based on a developmental model of this type of seeker, the processing bandwidth is very wide (~500 MHz) through the receiver function. Only the target discrimination and sensor control functions are amenable to digital techniques. However, the definition of these functions does not imply any advantage for digital processing, so only interfacing requirements at the subsystem level are applicable.



LA
MK
MX
XE

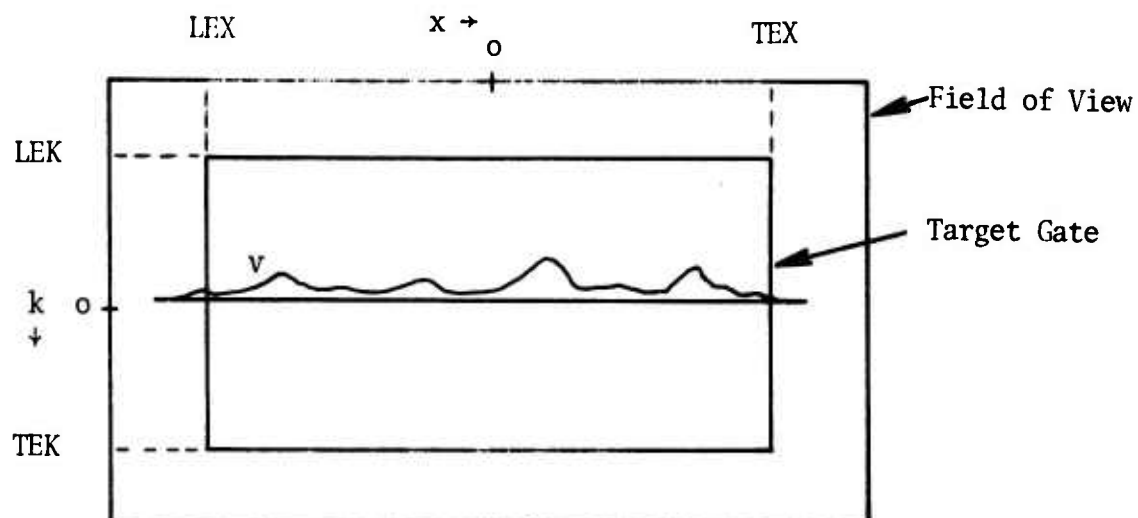
XS
CL
LE
EE
LB
TE

LINE INPUTS
$x_{11}, x_{21}, x_{31}$

LINE PROCESSING	
$LA = \sum x_{31}$	$A = A + LA$
$MK = MK + k \cdot LA$	$YE = YE + k \cdot LA$
$MX = MX + \sum x_{31} \frac{(x_{11} + x_{21})}{2}$	$\left\{ \begin{array}{l} LEK \leq k \leq EEK \\ LBK \leq k \leq TEK \end{array} \right\}$
$XE = XE + EEK - \min [EEK, x_{11}] + x_{21} - \min [x_{21}, LBK]$	

FIELD PROCESSING	
$XS = \frac{XE}{XS} \cdot KS + \frac{XS}{2}$	$YS = \frac{YE}{YS} \cdot KS + \frac{YS}{2}$
$CLX = \frac{MX}{A}$	$CLK = \frac{MK}{A}$
$LEX = CLX - XS/2$	$LEK = CLK - YS/2$
$EEK = CLX - XS/4$	$EEK = CLK - YS/4$
$LBX = CLX + XS/4$	$LBK = CLK + YS/4$
$TEX = CLX + XS/2$	$TEK = CLK + YS/2$

Figure 22. ATVS Tracking Algorithm

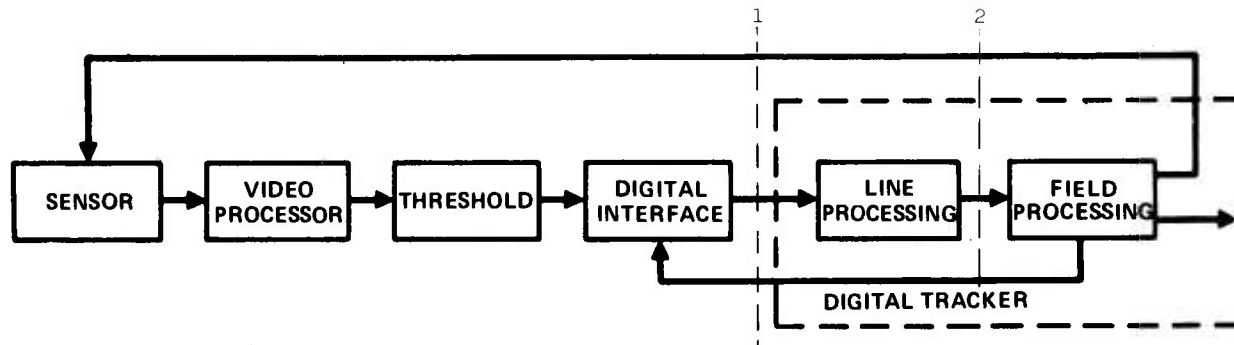


$$\left. \begin{aligned} \Delta A &= \sum_{i=1}^k V_i \\ \Delta X &= \sum_{i=1}^k V_i x_i \\ \Delta X1 &= \sum_{i=1}^k V_i x_i^2 \end{aligned} \right\} \quad LEX \leq x_i \leq TEX$$

$$\begin{aligned} A &= A + \Delta A \\ L &= L + k\Delta A & X &= X + \Delta X \\ L1 &= L1 + k^2\Delta A & X1 &= X1 + \Delta X1 \end{aligned}$$

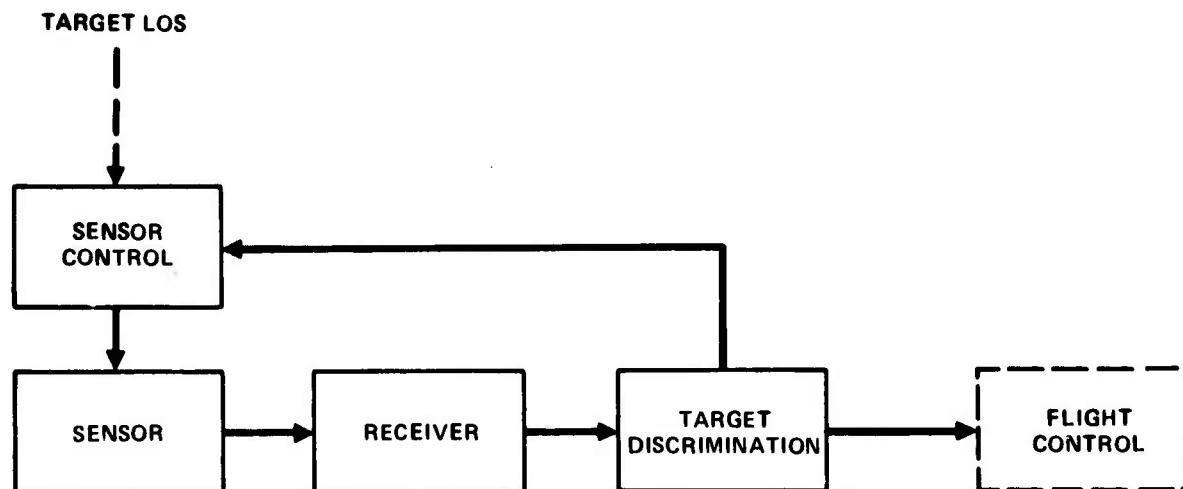
$$\begin{aligned} CLK &= L/A & CLX &= X/A \\ \sigma_K &= \sqrt{\frac{L1}{A} - CLK^2} & \sigma_X &= \sqrt{\frac{X1}{A} - CLX^2} \\ XK &= KG \cdot \sigma_K & XS &= KG \cdot \sigma_X \\ LEK &= CLK - XK & LEX &= CLX - XS \\ TEK &= CLK + XK & TEX &= CLX + XS \end{aligned}$$

Figure 23. Optical Tracker Algorithm



TRACKING ALGORITHM	PROGRAM SIZE	OPERAND MEMORY	THROUGHPUT		COMMUNICATION REQUIRED
			SHORT	LONG	
ATVS BASED INTERFACE ①	250	35	111 AT LINE 180 AT 60 Hz	5 AT LINE 11 AT 60 Hz	14 AT LINE 13 AT 60 Hz
ATVS BASED INTERFACE ②	200	35	10.8 KOPS	0.66 KOPS	18 AT 60 Hz
OPTIMAL INTERFACE ①	240	40	34 AT LINE 175 AT 60 Hz	2 AT LINE 18 AT 60 Hz	5 AT LINE 12 AT 60 Hz
OPTIMAL INTERFACE ②	205	40	10.5 KOPS	1.08 KOPS	17 AT 60 Hz
LINE = 15.75 kHz (EO), 4.8 kHz (IIR)					

Figure 24. Processing Requirements – EO/IIR Seeker



INPUTS: TARGET LINE-OF-SIGHT (2-3)

OUTPUTS: STEERING (2)

Figure 25. Radiometric Contrast Seeker Block Diagram

#### 4.2.3 Semiactive Laser Seeker Subsystem

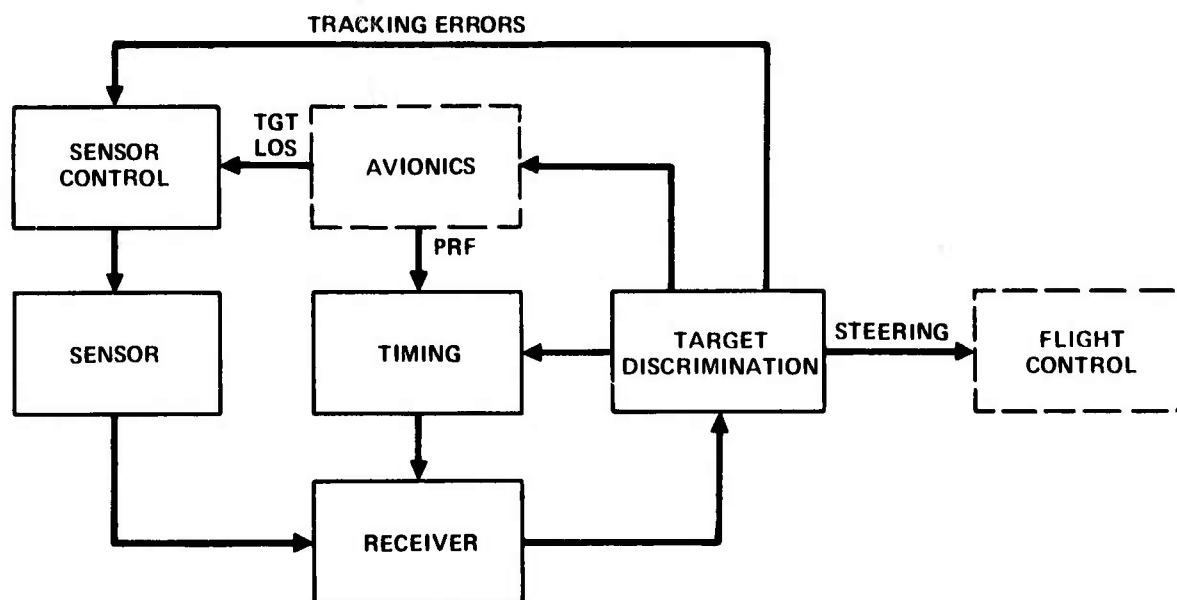
The functions of the semiactive laser seeker subsystem are shown in Figure 26. This subsystem operates in conjunction with an independent source of pulsed laser energy which illuminates the target. The initial conditions supplied by the avionics are the target line of sight and the pulse repetition frequency (PRF) of the laser illuminator. The sensor is a gimballed four-quadrant detector which is directed toward the target using the initial condition data. The outputs of the four quadrants are separately amplified in the receiver and then combined in a sum and difference network. The target discrimination operates on the sum output to detect the last pulse within a fixed gate at the designated PRF. The position of the target pulse within the gate is used to control the gate timing for the next pulse. The difference channel data at the target pulse time corresponds to the sensor pointing error information in the two axes and is used for both sensor pointing control and for steering the weapon.

The characteristic bandwidth of all processing through the target detection and pointing error derivation is approximately 50 MHz which is not compatible with current digital processing technology. The definition of the remaining functions does not imply any advantage for digital implementation, so only interfacing functions at the subsystem level are applicable.

#### 4.2.4 Anti-Radiation Seeker Subsystem

The functions of the antiradiation seeker subsystem are shown in Figure 27. This seeker operates on received RF energy and develops signals for weapon guidance to the designated radar source. Initial conditions supplied by the avionics designate the radar source characteristics — LOS, transmission frequency, PRF, pulse width, and intensity — to the appropriate seeker functions. Initially, the sensor is pointed along the target LOS, and the receiver is tuned to the designated frequency. Pulses at the desired frequency are amplified by the receiver and sent to the target discrimination function which selects the pulses from the designated source on the basis of PRF, pulse width, and intensity. Sensor pointing error information is developed to close the seeker control loop and for weapon steering. The source signal characteristics are updated using the measured pulse parameters both for improved discrimination and to aid in signal reacquisition, if required.

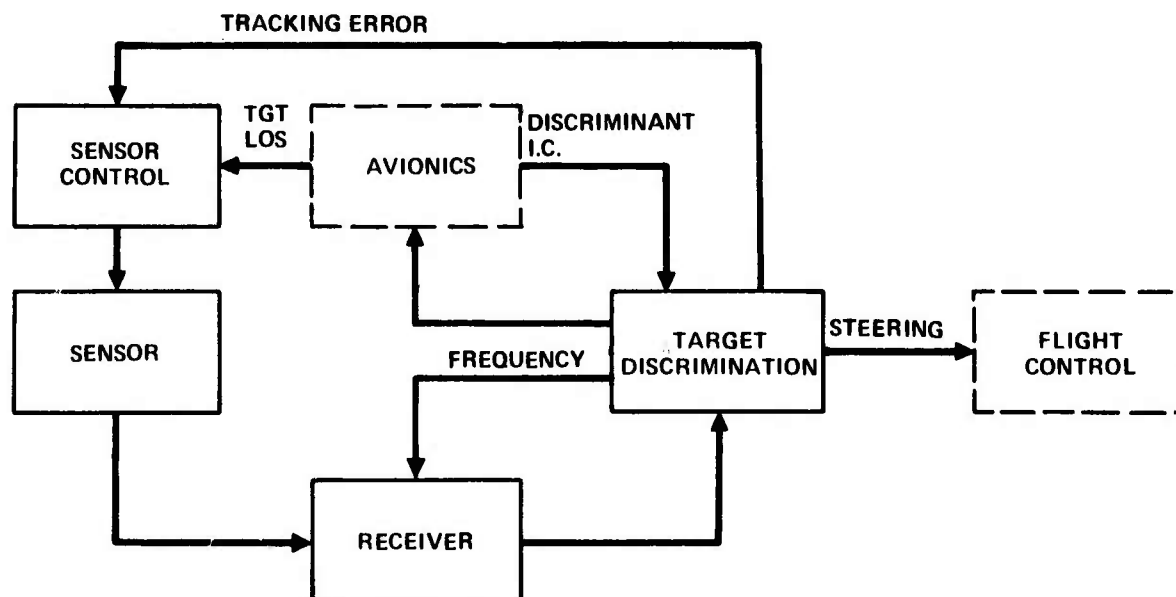
The target discrimination function was studied to determine digital processing requirements. The bandwidths of the sensor and receiver functions exceed digital processing capability, and some seekers of this type use body-fixed sensors implying that the sensor control function is really contained in the target discrimination function. The discrimination algorithms used in this study are based on the signal processing for HARM. The digital processing requirements in Table 8 vary with system level requirements on maximum input pulse density from which the desired pulse train must be discriminated. Consequently,



INPUTS: INITIAL CONDITIONS - TIMING, POSITIONING (2)

OUTPUTS: STEERING (2)  
LOCK-ON

Figure 26. Semiactive Laser Seeker Block Diagram



INPUTS: INITIAL CONDITIONS (6)

OUTPUTS: STEERING (2)  
LOCK-ON

Figure 27. Anti-Radiation Seeker Block Diagram



TABLE 8. PROCESSING REQUIREMENTS - ANTIRADIATION SEEKER

PROGRAM SIZE:	640
OPERAND MEMORY:	150
COMMUNICATION REQUIRED:	6 AT PRF 4 LOGIC AT PRF
THROUGHPUT:	INVALID PULSE REJECTION - $18 \text{ SHORT} / T_1^*$ VALID PULSE PROCESSING - $(135 \text{ SHORT} + 18 \text{ LONG}) \text{ PRF}$ REACQUISITION - $(75 \text{ SHORT} + 2 \text{ LONG}) \text{ PRF}$
* $T_1$ - ARRIVAL TIME DIFFERENCE, SECONDS, BETWEEN INVALID AND VALID PULSES.	
PRF - PULSE REPETITION FREQUENCY OF DESIRED PULSE TRAIN.	

throughput requirements are shown in terms of the worst case number of instructions which are executed for the processes involved in the discrimination function.

#### 4.3 FLIGHT CONTROL SUBSYSTEM

The functions of the flight control subsystem are shown in Figure 28. The two basic functions performed by the flight control are stabilization and steering. The stabilization function maintains airframe stability during weapon flight by deriving commands for the flipper actuators on the basis of measured angle rates of the airframe.

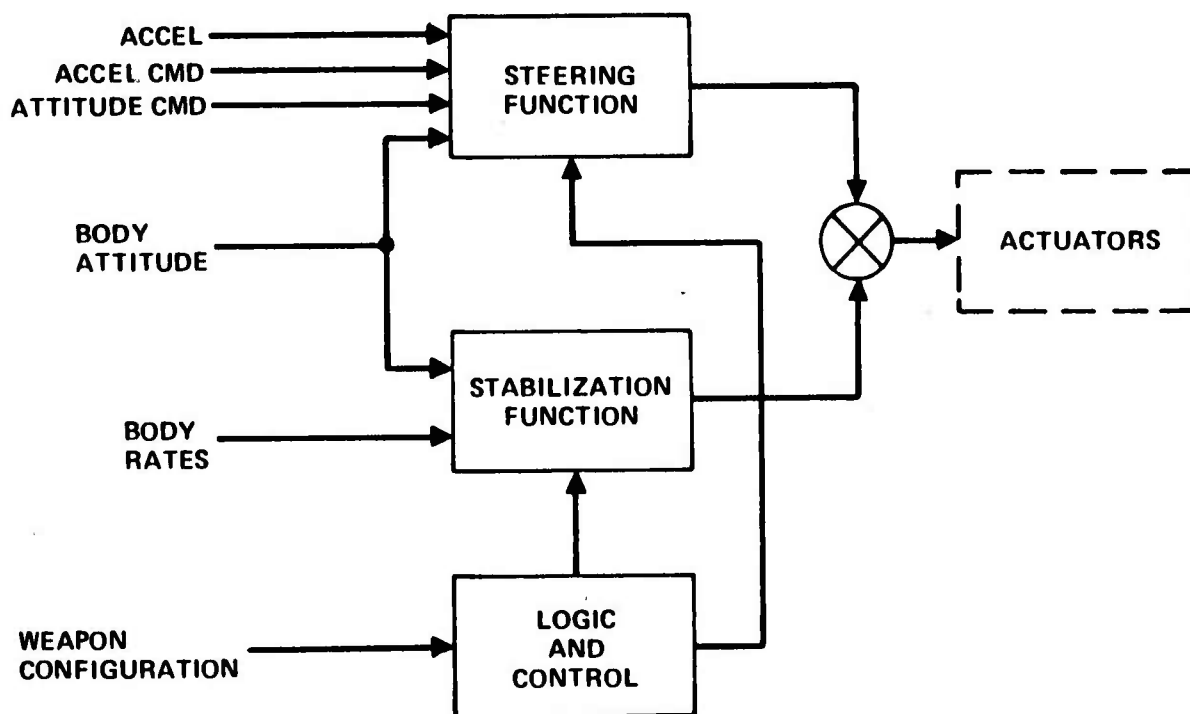


Figure 28. Flight Control Subsystem Block Diagram

The steering function operates on steering commands supplied by guidance subsystems as some combination of attitude or acceleration commands. The outputs of the steering and stabilization functions are combined to form commands to the actuator subsystems.

Considerable variations occur in both functions depending on weapon airframe configuration, phase of flight, and type of guidance subsystem. These variations require changes in both the form of processing to be performed and the processing parameters of the flight control subsystem. Flight control mode control and parameter selection is performed by the logic and control function shown.

The processing requirements for the flight control functions shown in Table 9 are based on the GBU-15 weapon system as implemented in PDAP. The PDAP requirements have been modified to account for differences in system requirements.

TABLE 9. PROCESSING REQUIREMENTS - FLIGHT CONTROL

FUNCTION	PROGRAM SIZE	OPERAND* MEMORY	THROUGHPUT		COMMUNICATION REQUIRED
			SHORT	LONG	
STABILIZATION	70	65	140 KOPS	8 KOPS	6 AT 400 Hz
STEERING	120	75	22 KOPS	1.8 KOPS	4 AT 50 Hz
LOGIC AND CONTROL	100	20	10 KOPS	0	1 AT 50 Hz
COMMON SUBROUTINES	400	**	**	**	0
ADDITIONAL AIRFRAME: <div><div>≤70 PM FOR STABILIZATION ≤120 PM FOR STEERING ≤100 PM FOR LOGIC</div><div>+ 100 CONSTANT MEMORY</div></div>					
*100 CONSTANT MEMORY PARAMETERS IN ADDITION TO DATA VARIABLE REQUIREMENTS SHOWN **INCLUDED IN ABOVE ENTRIES					

#### 4.4 ARMAMENT SUBSYSTEM

The armament subsystem consists of a fuze and a warhead. The processing requirements identified for this subsystem were parameter setting and status determination. A detailed examination of the requirements on these functions indicated no advantage for the implementation in the digital processor. Therefore, the only digital processing requirements for this subsystem are associated with data distribution.

#### 4.5 AIRCRAFT INTERFACE

The function of the aircraft interface is to provide mission-related parameters to the weapon subsystems. The type, number, and format of the parameters are strictly dependent on the weapon configuration, i. e., the component subsystems. Examples of these parameters have

been discussed for each subsystem in the previous paragraphs. In order to provide the appropriate parameters, the avionics subsystem must have knowledge of the weapon configuration. Several alternatives are available to provide configuration information for each weapon station on the aircraft: weapon control officer inputs (from mission plan), launcher data (set during weapon up-loading) and data directly from the weapon. Regardless of the method used, status data from the weapon are required to ensure that the avionics data have been correctly received and were the required data for the weapon configuration.

The digital processor functions associated with this interface are the formatting and distribution of data and control signals between the avionics and the weapon subsystem. The requirements for these functions are strictly configuration dependent, not only in terms of the weapon but also for the avionics, which may be either analog (existing avionics) or digital (DAIS, SMS). The stores management system (SMS) interface specification was used in this study to define both the functional and electrical interface parameters for digital avionics. It is expected that analog avionics may only be capable of operation with a subset of all possible weapon configurations, since some weapon subsystem parameters may not be available in the avionics system. These factors will be discussed in more detail in a later section.

## SECTION V

### POINT DESIGN FOR WEAPON CONFIGURATIONS

In Section IV, the role of a digital processor in a subsystem was discussed. For each of the applicable subsystems, several functional partitionings were made and the digital processing requirements corresponding to each interface were determined. Selection of the appropriate interface depends to some extent on whether the digital processor is just a subsystem component or whether it is a system component. In the latter case, other tasks are assigned to it and, for system or economic reasons, it may be appropriate to choose a different, less demanding interface than if it were a part of the subsystem.

The role of the digital processor in the total weapon system varies with the type of weapon system. Two system extremes are a fixed design weapon, such as Maverick, and a modular weapon with a complex array of possible subsystems. The latter type of weapon system is the object of this study. However, in order to gain an appreciation for the use of a digital processor in a modular weapon, it is instructive to examine its use in a fixed design.

In this section, three point designs are defined, corresponding to the three weapon configurations selected in Section II. Interfaces with each of the subsystems are chosen, and the digital processing requirements for the system are determined. A criterion for the selection of the interfaces was to maximize the amount of processing assigned to the digital processor subject to the capability of current technology. The result should be the most economical design for that particular weapon.

The three point designs arrived at in this section set the stage for a discussion of the role of a digital processor in a modular weapon, which is the subject of the following section.

#### 5.1 DESIGN GROUND RULES FOR POINT DESIGNS

The three configurations for which point designs are defined are listed in Table 10. The ground rules for the designs are:

1. Each design is to be separately optimized.
2. Within each design, the total processing load is to be partitioned between analog and digital with the object of maximizing the amount of digital processing subject to the present digital state of the art.
3. The processing assigned to digital equipment is further partitioned between special purpose digital equipment (assigned to the subsystem) and a central digital processor which exercises system software control.
4. There is a harness which connects the central processor to the subsystems. The link to each subsystem is dedicated (not shared with other subsystems) and is optimized for that subsystem.

5. All signal format conversions (e.g., analog-to-digital) are performed autonomously in the appropriate subsystem.

6. The subsystems have access to the processor through a hardware priority interrupt system.

7. There is no software executive. Task assignment in the processor is by way of the hardware priority interrupt system.

The throughput requirement arrived at for each of the systems is the average throughput which is the sum of the average throughput requirement for each task. This may be taken as the peak throughput requirement also if two assumptions are satisfied:

1. Propagation delay requirements for any particular function are compatible with the given throughput, considering the function by itself.

2. Any function with significant propagation delay requirements can be assigned a high enough priority to assure meeting its requirement.

If the assumptions do not hold, the given throughput requirements would have to be increased to accommodate propagation delay requirements.

TABLE 10. WEAPON CONFIGURATIONS

	CONFIGURATION		
	I	II	III
AIRFRAME	CRUISE	PWW	CWW
MIDCOURSE GUIDANCE	RAC/IMU	LORAN	DME
TERMINAL GUIDANCE	RAC/IMU	EO	IIR
FUZE	DIGITAL	DIGITAL	DIGITAL

## 5.2 WEAPON CONFIGURATION POINT DESIGN

### 5.2.1 Weapon Configuration I

The functional block diagram for this configuration is shown in Figure 29. The partitioning of the system is indicated by the letters in the right hand corner of each box. The rationale for the partitioning is based on the discussion of the subsystems given in Section IV. The throughput requirements for the given interfaces are also listed in that section. The combinations of the functions which are operative as a function of mission phase and the resulting throughput requirements are shown in Figure 30. The principal variations in processing configuration during weapon flight are in the parameter inputs to the navigation function. The weapon trajectory is controlled according to the position, cell size, and orientation of the stored reference array data. These reference data determine an appropriate aimpoint and approach angle for the weapon during flight. After the last position error data

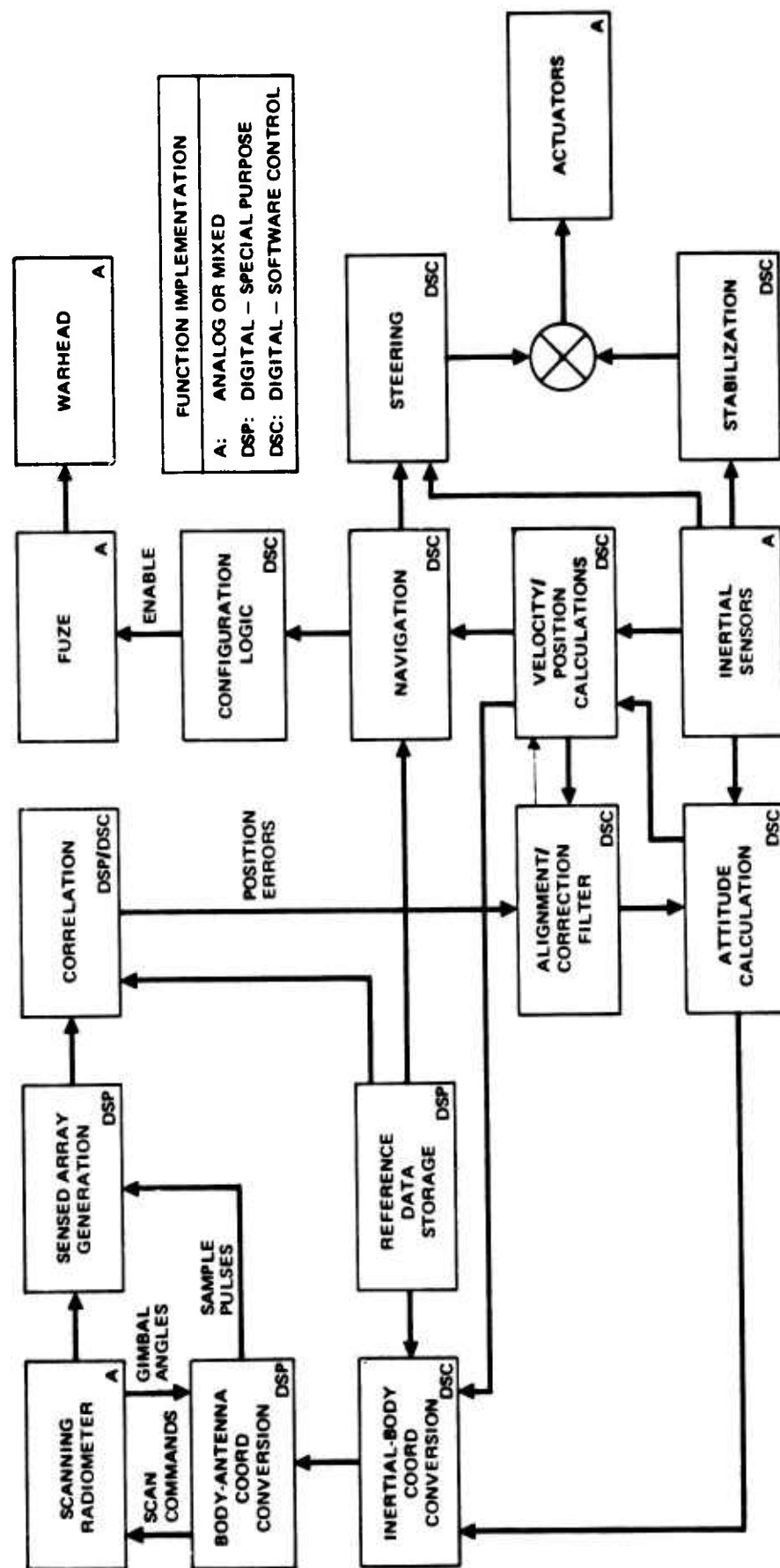


Figure 29. Configuration I Functional Block Diagram

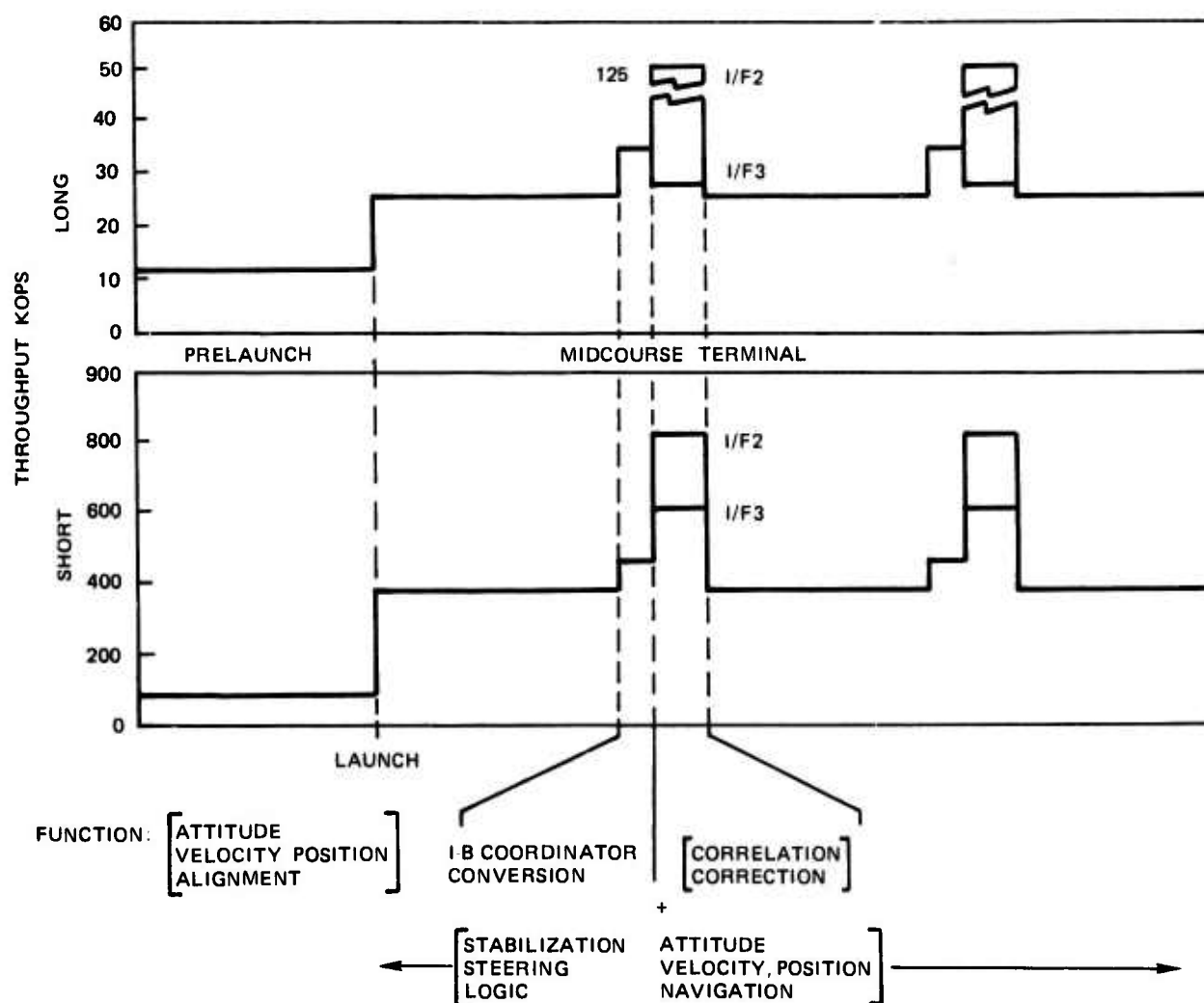


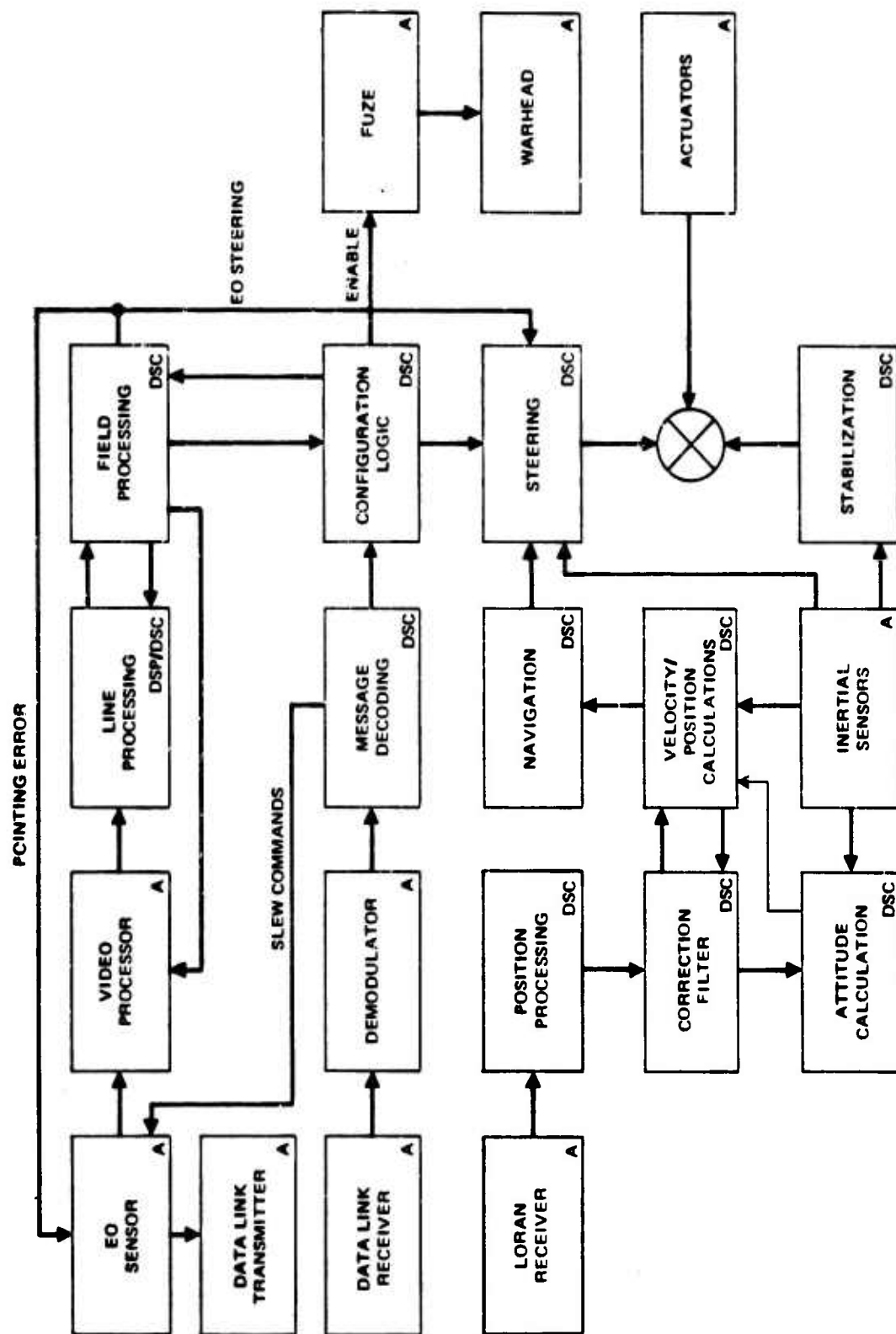
Figure 30. Throughput Requirements - Configuration I

are obtained, the aimpoint is set to the target coordinates, and the desired approach angle is set for best warhead effectiveness. Target coordinates may be input by the avionics during the prelaunch phase or may be input to the reference data storage during weapon assembly. The fuze is enabled at the appropriate weapon-target range.

### 5.2.2 Weapon Configuration II

The functions of this weapon configuration are shown in Figure 31 and the processor throughput requirements as a function of mission phase are shown in Figure 32. The target coordinates (avionics input during prelaunch) are used with the calculated weapon position in the navigation function for yaw plane steering during midcourse. The calculated LORAN position data are used to correct the weapon attitude, velocity, and position data during this phase.





**Figure 31. Configuration II Functional Block Diagram**



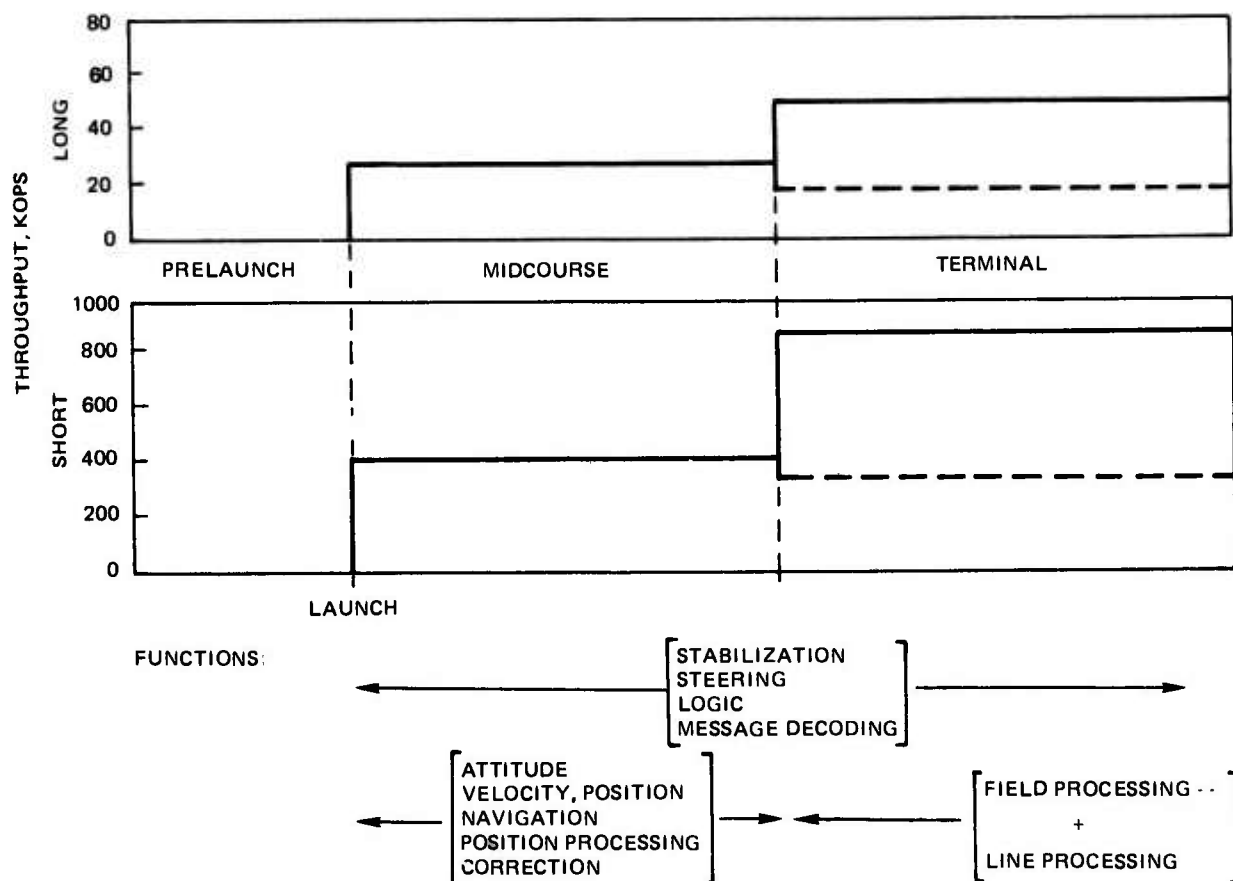


Figure 32. Throughput Requirements - Configuration II

The transition from midcourse to terminal is based on commands from the data link. In the terminal mode, the E-O steering signals are used for weapon guidance. The fuze is enabled by data link command.

### 5.2.3 Weapon Configuration III

The functions of this weapon configuration are shown in Figure 33 and the processor throughput requirements as a function of mission phase are shown in Figure 34. Midcourse guidance utilizes either data link (DL) or DME steering commands as determined by the status signals received from these subsystems (current GBU-15 guidance mode). The transition to terminal mode which uses the IIR steering signals for guidance is by command from the data link. The fuze is also enabled by data link command.

### 5.2.4 Configuration Requirements Summary

The processing requirements for each of the three weapon configuration designs are shown in Table 11. These requirements represent the worst case combination of functions during the typical mission for these configurations.

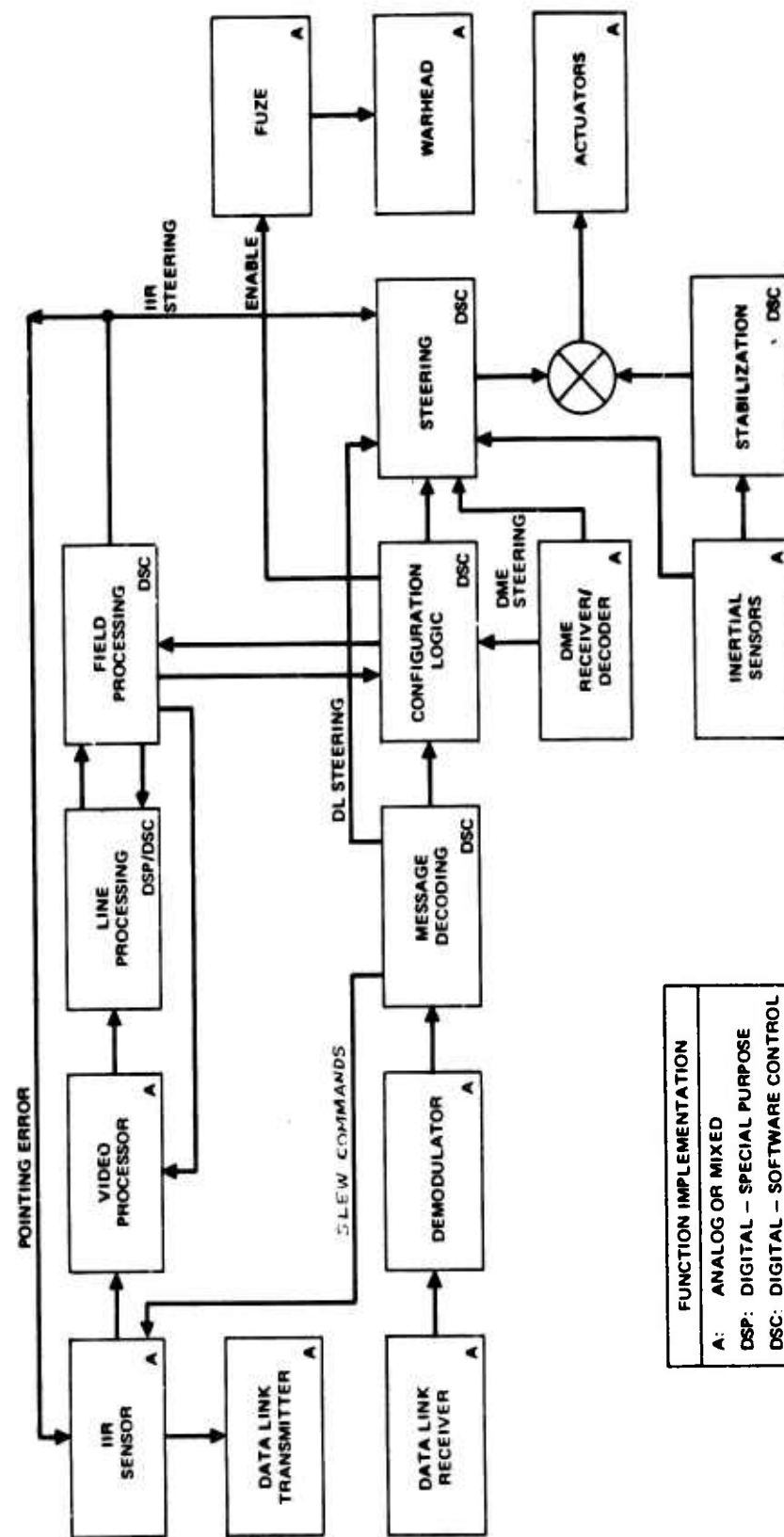


Figure 33. Configuration III Functional Block Diagram

TABLE 11. CONFIGURATION PROCESSING REQUIREMENTS SUMMARY

CONFIGURATION	PROGRAM SIZE	OPERAND MEMORY	THROUGHPUT, KOPS	
			SHORT	LONG
I	2540	850	814	125
II	2630	750	864	48
III	1030	320	492	26

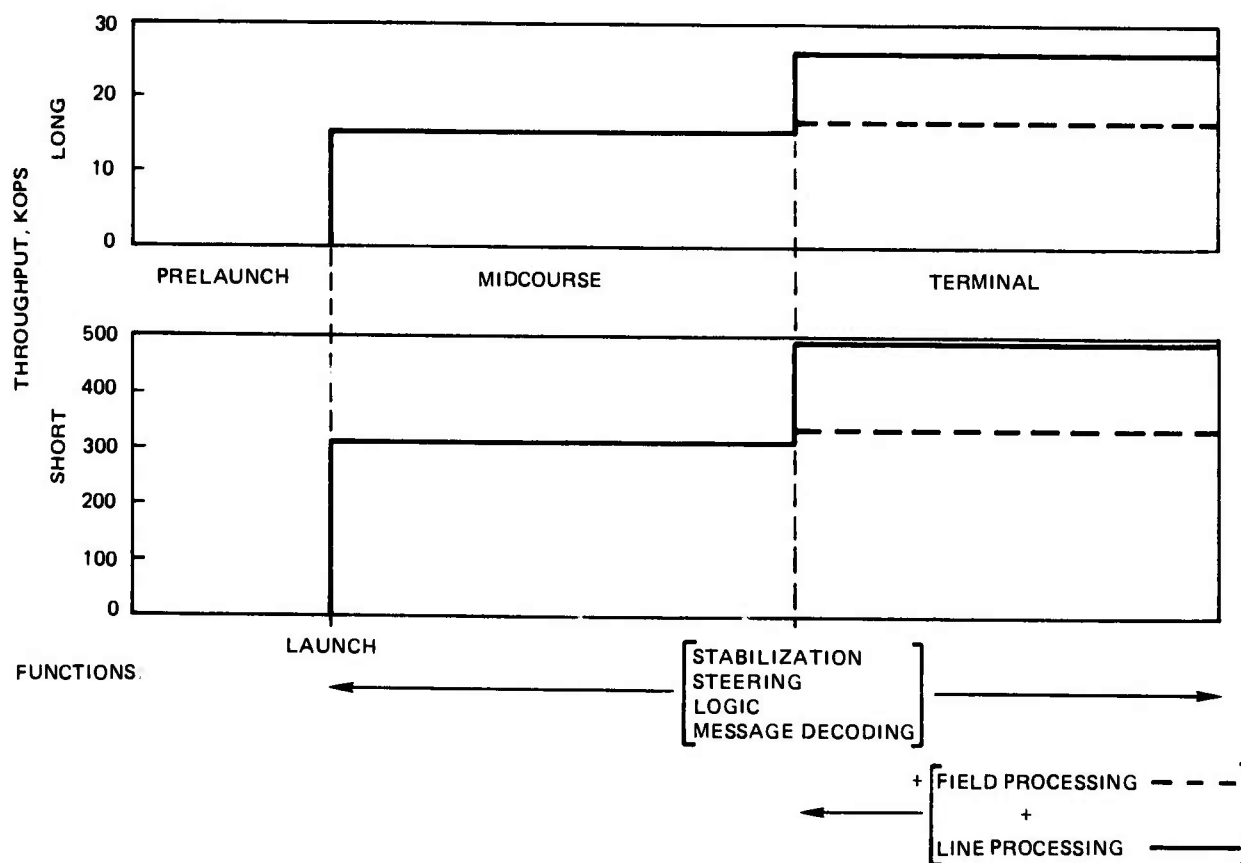


Figure 34. Throughput Requirements - Configuration III

It is noted that the requirements for the third configuration are much less than for the first two. In other words, a processor optimized for Configuration III would not be suitable for the other two configurations. However, a single processor could nicely satisfy the requirements for the first two configurations.

### 5.3 INTEGRATION IN POINT DESIGNS

Up to this point, an integration function in the point designs has not been explicitly considered. It is clear that the digital processor

has a key role in the integration. It not only does some of the computations for the various subsystems, but it also distributes the results to the appropriate place. One can consider the harness and the processor together as the integration subsystem for the weapon. The harness provides the physical link to each of the subsystems and the processor is the nexus. It functionally links the subsystems together into a working system, i.e., it provides the interface between all the subsystems. The functional interface between the processor and the various subsystems is summarized in Table 12, which lists the signals entering and leaving the processor. The interface signals are categorized by format (analog versus digital) and data rates are shown for each type. The variations in the number and data rate of each type of interface signal over the three configurations are quite apparent. These variations are a result of both differences in the type of functions performed in the three configurations and differences in requirements for the equivalent functions.

TABLE 12. INTERFACE SIGNALS FOR POINT DESIGNS

SIGNAL TYPE	WEAPON CONFIGURATION		
	I	II	III
ANALOG INPUTS	3 AT 400 Hz 1 AT 10 Hz	3 AT 400 Hz 1 AT 10 Hz	3 AT 400 Hz 4 AT 50 Hz
ANALOG OUTPUTS	3 AT 400 Hz	3 AT 400 Hz 2 AT 60 Hz 2 AT 30 Hz	3 AT 400 Hz 2 AT 60 Hz 2 AT 30 Hz
DIGITAL INPUTS	3 AT 10.9 kHz 3 AT 100 Hz 1 LOGIC AT 100 Hz 8 AT 1 Hz	5 AT 15.75 kHz 6 AT 100 Hz 2 AT 30 Hz 2 AT 1 Hz	5 AT 4.8 kHz 2 LOGIC AT 50 Hz 2 AT 30 Hz
DIGITAL OUTPUTS	28 AT 100 Hz 1 LOGIC AT 50 Hz	8 AT 60 Hz 1 LOGIC AT 50 Hz	8 AT 60 Hz 1 LOGIC AT 50 Hz
INTERRUPT INPUTS	1 AT 10.9 kHz 1 AT 400 Hz 1 AT 100 Hz	1 AT 15.75 kHz 1 AT 400 Hz 1 AT 100 Hz 2 AT 30 Hz	1 AT 4.8 kHz 1 AT 400 Hz 1 AT 30 Hz

## SECTION VI

### THE MODULAR WEAPON AND INTEGRATION

The major purpose of this study was to determine the role of digital processing in integrating the components of a modular weapon system. In the preceding section, the part played by a digital processor in integrating the components of a fixed-design weapon was described. In the latter situation, the integration subsystem is a combination of a digital processor and a harness with dedicated links to the subsystems. It is the purpose of this section to determine the corresponding integration subsystem for a modular weapon, and to determine what functions must be assigned to the digital processor to accomplish integration. The first step is to examine the characteristics of a modular weapon system.

#### 6.1 MODULAR WEAPON CHARACTERISTICS

The salient feature of the modular weapon is that the subsystem of the fixed-design weapon is replaced by a generic subsystem. In the assembly process, there are points where a choice is possible. For example, it is not the terminal guidance section, but one of N terminal guidance sections. The fixed interface of the fixed-design becomes a variable interface, at least in functional detail. The real integration problem of the modular weapon is to control this variable interface.

##### 6.1.1 Description of Modular Weapon (Weapon Assembly Level)

It is assumed that the weapon system is stored, (in the forward area) as sections which are assembled into weapons as the need arises. An assembled weapon might be described as

$$A_1 - B_3 - C_1 - D_1 - E_2 - F_2.$$

This is interpreted as a weapon with the nose section of the first type ( $A_1$ ), the second section of the third type ( $B_3$ ), and so on. It is not required that each spot in the sequence be filled for all weapon configurations. The modularity concept requires that

$$A_1 - B_3 - C_1 - E_2 - F_2$$

might be a weapon configuration also. The deleted segment could be a propulsion module. Also required by the modularity concept is some degree of interchangeability of the section sequence. For example, weight and balance considerations may require a weapon like

$$A_2 - B_4 - D_2 - C_2 - E_2 - F_2.$$

Also admitted into the modularity concept are two other kinds of constructions; bolt-ons and slip-ins. A bolt-on is a module that bolts on to one of the sections. The purpose could be to modify the aerodynamics

or to add propulsion. In the above notation, a bolt-on fastened to section  $D_i$  would be designated as

$$\begin{array}{c} D'_j \\ - D_i - \end{array}$$

which is interpreted as the  $J^{\text{th}}$  choice of bolt-on for section  $D_i$ .

A slip-in is a subsystem which is optionally placed in a section. It is thought that, in general, sections will be complete as delivered to the forward area. However, it allows a bit more flexibility in the modularity concept if slip-ins are allowed. An example of a slip-in might be an altimeter which would be required in only a few configurations. In the above notation a slip-in would be designated by an inferior letter:

$$\begin{array}{c} - D_i - \\ D''_j \end{array}$$

where  $D''_j$  represents the  $J^{\text{th}}$  choice of a slip-in for the  $D_i$  section.

In summary, then, the modularity concept used in this study (expressed at the section level) allows constructions like

$$\begin{array}{c} D'_1 \\ A_1 - B_1 - C_3 - D_2 - E_4 \\ C''_1 \end{array}$$

with the possibility of interchanging section positions. That is,  $C_i - D_j$  can become  $D_j - C_i$ .

The spirit of modularity demands that such constructions be accomplished with a minimum use of configuration - specific adaptor modules or adjustments. That is, it is undesirable to require the assembler to have to select special adaptors for specific sections or subsystems. It is also undesirable to require that special adjustments or parameter settings be made on selected sections.

Another way of describing the modular weapon is by the subsystems or functions it contains. The generic subsystem list contains such items as

- Warhead
- Airframe
- Propulsion
- Inertial sensors

- Flight control
- Midcourse guidance
- Terminal guidance.

Each of these generic subsystems can have several specific choices. In general, there is a close relationship between the subsystem description and the section description for the weapon. For example, almost always the terminal guidance subsystem will be located in the forward section. However, the modularity concept used in this study allows both kinds of exceptions. An exception of the first kind is to allow a section's usual role to change. For example, the forward section, which usually contains the terminal guidance function, may sometimes contain another function (e.g., midcourse guidance sensor). An exception of the second kind allows a particular subsystem to be located in different sections in different configurations. While these exceptions will probably be rare, they are admitted to the modularity concept to allow greater flexibility. The spirit of modularity demands that these exceptions do not require any special adjustments or parameter settings in the assembly process.

#### 6.1.2 Growth Considerations

It is desired to allow additional specific subsystems to be added to the weapon system (or changes made to an existing subsystem) without extensive rework on existing equipment. For example, suppose that a new terminal guidance subsystem is added, and it is completely contained in the nose section. The modularity concept requires that no changes will have to be made in the other sections; this includes harness modifications.

It is also desired that field retrofits be accomplished with a minimum of rework to existing sections. Ideally, a new box replacing an existing box in an existing section should fit in the same physical location and use the same harness termination even if the function of the box has changed.

The modularity concept described above is the basis for defining the integration subsystem of which the digital processor is a part.

### 6.2 THE INTEGRATION SUBSYSTEM

The integration subsystem for a fixed-design weapon, as noted previously, is the combination of the digital processor and a harness with dedicated links to the subsystem. Thus, each subsystem has its own dedicated communication link with the processor. On this link the subsystem can request data from the processor or tell the processor that it has data ready. This capability is furnished by an interrupt feature in the processor. The interrupt capability must be vectored and each subsystem requires its private line. Transmission of data from a subsystem to the processor requires a private line for each discrete and a serial or parallel link for general data. Discretes will terminate in the equivalent of a register (flag input). More general



data transfer is via some kind of shared memory. Transmission of data from the processor to the subsystem is by the same means (flags or shared memory).

In the point design, the physical layout can be adjusted to subsystem needs. For example, if one of these dedicated links requires a path for high speed parallel data, the physical arrangement in the point design can be adjusted to make that part short. In general, special accommodations can be made for each path to account for peculiar requirements of the subsystem.

The harness concept in the fixed-design weapon clearly violates the modularity concept. In the first place, the dedicated harness does not furnish a common interface at the section level. The harness carries only the lines required for the following sections. In order to provide a common section interface with this type of harness, all lines for all functions would have to pass through all sections. This bus structure is impractical for a large number of lines, and the number will be larger, particularly when growth provision is included. Furthermore, the harness provides a unique interface for each subsystem. This situation could be tolerated for a given generic subsystem in a fixed position. In this case each species of the subsystem genus could be required to furnish a common interface. However, since in this type of harness each wire has a specific meaning, one is limited in the kinds of information that can be exchanged with the subsystem. This limits growth. Moreover, in the modular weapon concept, a particular harness link terminus does not always attach to the same generic subsystem. In the dedicated harness concept, extra dedicated links would be required to account for these cases.

For the above reasons, a harness with dedicated links does not appear to be a good choice for the modular concept. Some type of bus is required to give the common interface at section level and subsystem level. To limit the number of wires in the bus, it should be time shared among the subsystems (i. e., time multiplexed). Time sharing among the subsystems does not necessarily require a common format for data on the bus as long as there is a way of identifying data source. However, since a given entry point on the bus does not necessarily correspond to a particular generic subsystem, it would appear that a common format is required.

The bus structure must provide all the functional features of the dedicated harness. In particular, it must provide a high enough data rate to accommodate the bandwidth of all required data. By required data is meant the data necessary for the integration function. Also, the bus must provide an interrupt feature to allow subsystems to signal the processor that access is required. The bus should also provide for signaling and control functions (as well as data) from the processor to the subsystem. Among the possible formats for the bus are:

1. A group of lines on which analog signals may be placed
2. A serial, digital format

3. A parallel digital format
4. Some combination of the above.

Some of the data which must be transferred on the bus require a precision greater than given by an analog system (e.g., inertial reference data); therefore, it is necessary that the bus include a digital format. Including an analog format, as well, would allow centralizing the analog-to-digital and digital-to-analog conversion for some of the signals. However, transmission of analog signals over a long bus is a much more difficult process than for digital signals because of voltage offsets and noise pickup. Therefore, it was decided to limit the bus to a digital format. The selection of a serial or parallel format will be made after a discussion of the functions to be transmitted on the bus.

In summary, the integration subsystem consists of the digital processor and a digital, time-multiplexed bus. The bus traverses each weapon section and provides a common interface at the section level. Interior to the section, bus spurs tap off from the bus to provide a common interface at the subsystem level. It should be noted that the interface at the section level is not necessarily the same as the interface at the subsystem level. In other words, the transmission format on the bus is not necessarily the same format as presented to the subsystem.

The bus provides interrupt capability as well as data transmission capability.

### 6.3 PROCESSOR FUNCTIONS REQUIRED FOR INTEGRATION

As stated before, the primary function of the digital processing system is weapon integration. In Section V, three point designs were presented in which the processor did more tasks than just required for integration. In fact, one of the objectives of that exercise was to maximize the number of tasks performed by the processor. At this point we wish to determine just those tasks which are required for the integration function. It is these tasks which will be the basis for estimating the required processing system capabilities.

#### 6.3.1 System Management Functions

The system management functions are those tasks necessary to combine the separate subsystems into a working system. In the fixed-designs considered earlier, much of this function was performed by the dedicated harness and such features as direct, hardware, vectored interrupt in the processor. These features do not exist in the integration subsystem described in subsection 6.2. In the modular weapon, the system management tasks are software controlled and performed by the digital processor. These tasks are described in the following paragraphs by a step-by-step analysis of what the processor has to do for this integration function.

The first step in the integration process is for the processor to determine the weapon configuration. In order to do this, it is necessary that each subsystem which can be a variable in a configuration identify

itself to the processor. This requires communication over the bus. The control of the bus communication is a task which was present in only rudimentary form in the fixed-design weapon. In the modular weapon, with the time multiplexed bus, it is a much more significant function. This task is also assigned to the processor. The subsystem identification process, incidentally, places a definite requirement on each subsystem or functional module to provide an identifier on request.

After configuration identification, the processor must initialize the system by setting parameters using a combination of pre-stored data and mission-related data from the avionics. It should be noted that the processor considers the interface with the aircraft avionics as another subsystem as far as the above tasks are concerned.

After initialization, the processor exercises control over the sequence in which missile tasks are performed and controls the required communication on the bus. This is a real time control function. The throughput of the combined bus-processor integration subsystem must be high enough to satisfy any propagation delay requirements of the subsystem.

The weapon may also have to go through a self-test sequence, either while on the aircraft or before it is loaded onto the aircraft. The self test sequence is also controlled by the digital processor.

The above-described functions are called system management functions and are a necessary part of integration. They are listed below.

- Configuration Identification
- Communication Control
- Initialization
- Sequencing
- Self-Test

### 6.3.2 Flight Control

Another function which is closely related to integration is the flight control, which has two subfunctions: stabilization and steering. Stabilization, from the viewpoint of a digital autopilot, is a cascade of filters which operate on inertial sensor data to form flipper commands to stabilize the weapon in flight. The required arrangement of the filters is a function of the aerodynamics. The flight control function integrates the proper total filter function from available modules on the basis of logic signals giving the status of missile configuration.

In a similar sense the steering part of the flight control function synthesizes the proper steering commands to put into the autopilot steering loop on the basis of the missile configuration and trajectory status.

The flight control function is thus truly a weapon integration function in the modular weapon.

### 6.3.3 Strapdown Inertial Reference

There is one more function which should be analyzed with respect to weapon integration. This is the strapdown inertial reference function (SIRF). In all the intended uses in the weapon system analyzed in this study, the SIRF is used in conjunction with another subsystem (a mid-course guidance sensor) which provides position and/or velocity update information for the SIRF. There are several different update subsystems (RAC, GPS, LORAN) and the information from each is filtered differently for use in the SIRF. Thus, the complete SIRF, including the proper filtering of the appropriate update data, is indeed an integration function.

### 6.3.4 CORE Functions

The above integration functions, System Management, Flight Control and Strapdown Inertial Reference, are the basic functions which the digital processor must perform. These CORE functions have been used to set the requirements for the DP and the bus. As pointed out later, when the peak throughput requirements for the CORE functions, including propagation delay requirements, are satisfied, the digital processor has capacity to do other, lower priority, tasks also. The tasks which might be included are typically different in different configurations. Some of these will be discussed in the following section.

### 6.3.5 Bus Transmission Requirements for CORE Functions

At this point we are in a position to make a tentative decision as to whether the bus should be a serial or parallel structure. The processing and bus rate requirements for the system management functions have not yet been derived; however, we do have the data from the three fixed-point designs. The required bus rates for these designs is shown in Table 13.

The highest transmission requirement shown in the table is about 100 K words per second if one includes the requirements for both data and interrupts. The data words require 16 bits for data and perhaps 4 bits for control purposes. Thus the required bus bit rate is about 2 million bits per second. This rate is normally considered too high to support with a shielded, twisted pair bus such as would probably be used here. However, this refers to buses with lengths of hundreds of feet or more. The bus in the modular weapon will probably not exceed 20 feet in length. During the course of this program, transmission measurements were made on a 6 meter bus with eight terminals. Transmission rates up to 10 million bits per second were achieved. Thus, the 2 million bits per second should not present any problem.

Now the data in the table refer to a somewhat different case than we are talking about for integration. The Configuration II fixed-design included the flight control and the strapdown inertial reference function, but not all of the system management functions. It did include, however, line and field processing for the E-O seeker and position processing for the LORAN system. The E-O processing requires higher bus rates than estimated for the system management function. Therefore, the rates required for Configuration II exceed the rates required

TABLE 13. BUS TRANSMISSION REQUIREMENTS FOR FIXED-DESIGN WEAPON CONFIGURATIONS

FUNCTIONS	TRANSMISSION REQUIREMENTS (words/sec)		
	CONFIGURATION		
	I	II	III
DATA AND CONTROL	39,000	83,000	28,000
INTERRUPTS	11,400	16,300	5,300

for the CORE functions. The conclusion is that a serial bus is adequate for the integration subsystem. This conclusion is verified in the next section which deals with system design.

#### 6.4 CORE DIGITAL PROCESSING SYSTEM

The overall characteristics of the integration subsystem for the modular weapon were determined in this section. This subsystem is called the digital processing system for the CORE (integration) functions. Figure 35 illustrates the system. The serial digital bus, called the weapon bus, provides both data transmission and interrupt capability.

This bus provides a flexible communication and control medium for integration of the weapon subsystems. The standard interface which is presented to all subsystems is digital, and signal conversion equipment is required for compatibility with existing subsystems as shown in the figure. All data transmissions are controlled by the digital processor software, while interrupt transmissions may be initiated by any subsystem. The CORE functions performed in the digital processor are System Management, Flight Control, and Strapdown Inertial Navigation.

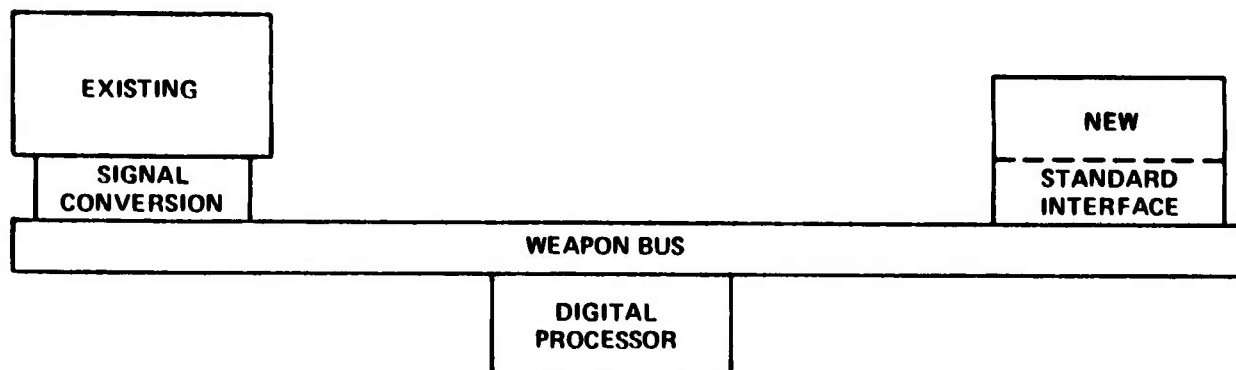


Figure 35. Digital Processing System Configuration

## SECTION VII

### DIGITAL PROCESSING SYSTEM DESIGN

Having determined the broad features of the digital processing system, it is now possible to proceed with a more detailed definition and to derive performance requirements. While a multiplexed digital bus has been specified, the bus configuration is still an open question and tradeoff studies are necessary to select the preferred configuration. Likewise, the processor configuration has not yet been determined.

In this section, the tradeoff factors pertinent to bus and processor configuration are examined in order to make a configuration choice.

With the overall processing configuration in hand, it is possible to proceed to a definition of the software structure which will support the system modularity requirements, and then to specify processor architecture and performance. These subjects are treated in the following paragraphs.

#### 7.1 WEAPON BUS DESIGN

The principal tradeoff areas in the weapon bus design are the bus topology and bus control methodology. In the previous section, a bit serial digital transmission format was shown to be compatible with the transmission rate requirements of the weapon system. Bus topology is concerned with the structure of the weapon harness which provides all required functional interconnections between the weapon subsystems.

Bus control is concerned with the resolution of bus usage conflicts in the real time operating environment. These conflicts arise because of the asynchronous nature of the operations of the weapon subsystems, i.e., not only the frequency but also the phasing of the intersubsystem communication requirements are determined by each subsystem.

Two general types of intersubsystem communication and control philosophies can be considered for this weapon system. The first philosophy would require each subsystem to output all of its data and status information each time it is updated within the subsystem, regardless of whether the information is needed by the other weapon subsystems. All weapon subsystems would examine all messages and accept the parameters which it needs to perform its functions. There are obvious problems with this philosophy. First, a system wide identifier must be assigned to each subsystem parameter to facilitate the decision process in the other subsystems. Second, the subsystems must contain decision circuitry and identifier storage for all pertinent input parameters. The bus capacity must be sufficient to allow for both required and useless parameter transmission. A final problem with this philosophy is that data consistency cannot be guaranteed. Since each subsystem outputs its data asynchronously when ready, the data in the user subsystem may be only partially updated when needed by the user. This philosophy leads to implementation complexity which is undesirable.



A second philosophy implies that only pertinent parameters are output by each weapon subsystem in response to the requirements of the other weapon subsystems. This type of system communication control requires the transmission of words indicating both the need for data and what data are required. This is the function of the interrupt words discussed in the previous section. The total bus traffic is obviously lower in this case since only required parameters are transferred between the weapon subsystems. However, the problem associated with system-wide identification of both subsystem parameters and interrupts still exist. This problem will be addressed in subsequent paragraphs of this section.

Weapon configuration information is contained in the digital processor software which can be used to determine the required intersubsystem communication for the weapon. Two real time bus control philosophies may be considered: central and distributed. The distributed control philosophy requires that weapon configuration information be sent to each subsystem to minimize the complexity of the bus control circuitry in the subsystem. Central control by the digital processor allows minimum complexity in each subsystem. The applicability of these control philosophies is presented for different bus topologies in the following paragraphs.

#### 7.1.1 Bus Topology and Control Structure

Ideally, the bus topology should allow modular addition or deletion of subsystems within a weapon section without modification of the weapon harness associated with the bus. There are two topologies which provide this simplicity in weapon assembly. These are the ring and serial structures shown in Figure 36. Both structures can provide the required intersubsystem communications, but their implications on subsystem requirements must be determined. Other topologies such as the star, fully connected, and tree structures require that new signal paths be added to the existing structure whenever a new module is added.

##### Ring Structure

The ring structure utilizes a distributed control philosophy and allows each subsystem to output both data and control parameters, and interrupt words asynchronously on the data bus. The ring operates by transmitting a message word from one unit to the next in sequence around the ring. A given station can transmit a word only to a station adjacent to it and on its right hand side. Each word must carry an address identifying its destination. If a station receives a word that is not addressed to it, it transmits the word in the ring at the next word time. If there are  $N$  stations on the ring, there can be as many as  $N$  words being transmitted at a time, but it may take  $N-1$  word times for a message to get from its source to its destination. The control of the ring structure is relatively simple since any subsystem can put a message word on the bus when it detects an empty word slot in the ring. This provides good time response in an asynchronous operating environment provided that empty slots are available. However, the timely existence of empty slots in the structure cannot be guaranteed



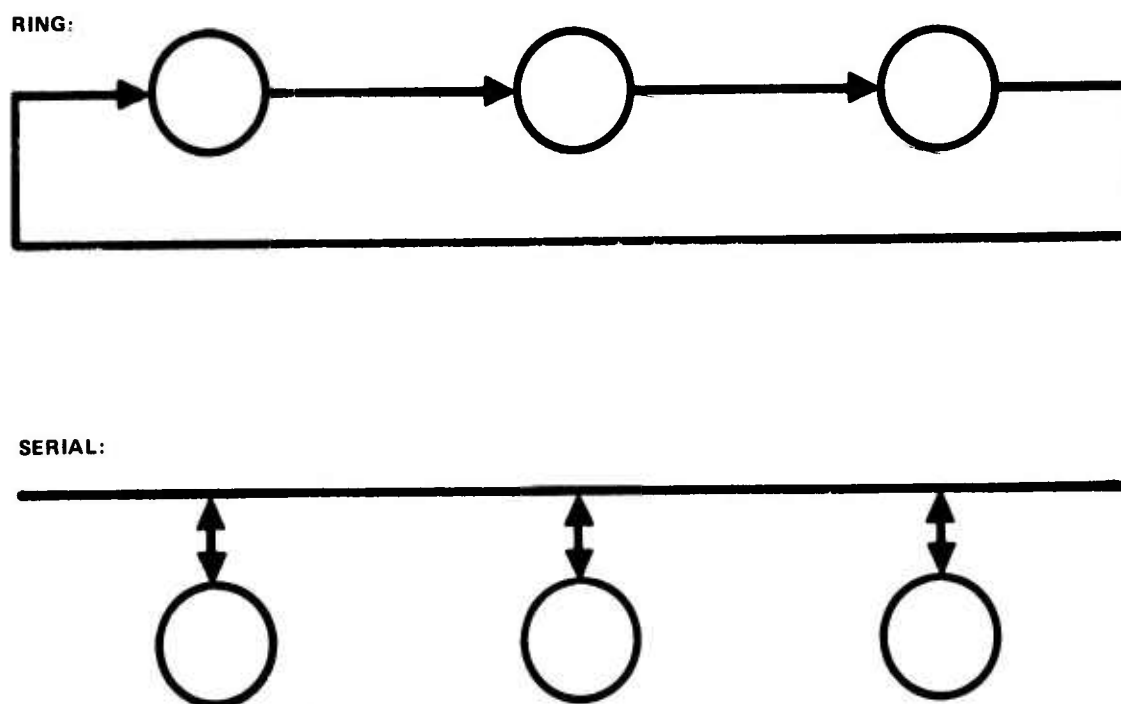


Figure 36. Weapon Bus Topology

for time critical parameters in all weapon configurations. The total bus transmission delay also depends on the number of subsystems in the weapon configuration between the source and destination of the transfer.

Empty slots can be created for time-critical parameters by removing other data from the ring, storing the data, and then putting it back on the bus after the critical parameters have been transferred. However, this implies that sources of time-critical data must be capable of assessing the priority of their data relative to other data in the system. This requirement introduces complexity into the component subsystems.

The priority problem could be solved also by providing sufficient bus capacity to transfer time-critical data in the worst case operating environment. However, it is nearly impossible to identify a worst case environment within the current weapon system definition, without considering the addition of subsystems in the future. In addition, each word transferred must have a companion acknowledge word from the destination to indicate correct reception of the word, which reduces the actual bus capacity by 50 percent. Each source subsystem must store all output data until the appropriate acknowledge is received to allow retransmission in case of error. As noted before, each word on the bus must contain the destination. In addition, the data must be identified in the word to determine its use in the destination subsystem. The requirement for both a destination address and data identifier in each word results in a large number of bits in each message word relative

to the data content of the word. This type of overhead is present for every word of a block transmission since consecutive transmission of the words in a block transfer cannot be guaranteed in the ring structure. Message word format will be discussed in a later paragraph.

### Serial Structure

The serial structure operates by having all stations on the bus simultaneously receive the transmitted message. Each station must decide for itself whether or not the received transmission is for it. Only one word may be in transit at a time, but a word may be sent from any source to any destination in a single word time. Correct reception of a word is easily acknowledged since it pertains to the last word transferred on the bus.

The control of the serial structure is more complex than for the ring structure, but the same factors discussed previously for the ring structure also apply to the serial structure. The control problem has two parts: determination of whether the bus is in use, and resolving priority conflicts in putting messages on the bus. Three candidate control structures were investigated for the serial bus: rotating window, bus race, and central control. Both the bus race and rotating window structures provide distributed real time control of the bus. These structures provide for transmission of data and control parameters, as well as interrupts on the data bus.

The rotating window structure shown in Figure 37 is very similar to the ring structure previously discussed in that permission to put messages on the bus is passed from one subsystem (DE) to the next in sequence. In fact, if the window is only one message word long, the serial structure essentially degenerates to the ring structure except for allowing messages to be transferred between any two subsystems in one word time. If the window may be retained by a DE until all words of a block transmission are transferred, some increase in bus efficiency is allowed by appropriately formatting the message words. However, this philosophy would allow a low priority block transmission to prevent transfer of time-critical data by another DE. Since this control technique does not allow priority assignment for bus messages, it is not recommended.

The bus race control structure is shown in Figure 38. The bus race structure operates by connecting the DEs with a single control line labeled the message in progress (mip) line. Whenever a DE is transmitting a message, it also places a level on the mip line. Before starting a transmission a DE will inspect the mip line to assure itself that the bus is not in use. If the bus is in use, the DE wishing to make a transmission will monitor the mip line until the level disappears. It will then raise the mip line itself and begin transmitting.

A problem arises when two DEs wish to make a transmission simultaneously. Following the procedure just outlined, both units would wait for the mip level to disappear and then both would race for

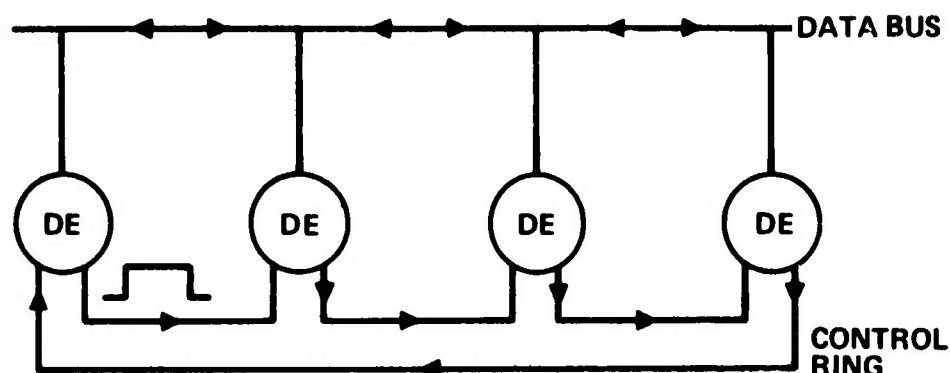


Figure 37. Rotating Window Control Structure

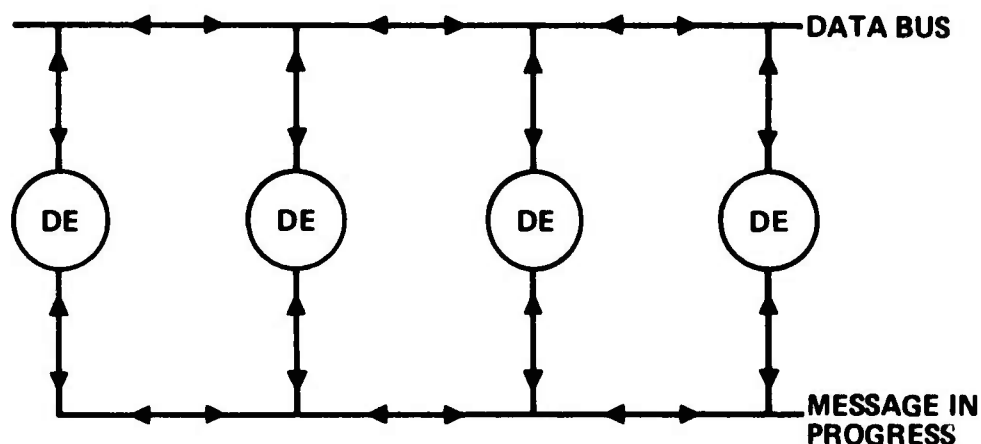


Figure 38. Bus Race Control Structure

the bus. The outcome of the race would probably be that both units would think they'd won and start transmitting simultaneously. This would result in erroneous transmissions on the bus.

To avoid this problem, a slight modification is required. Each unit is assigned a fixed wait period. When a DE wants to transmit, it first raises the mip line, waits its assigned period and then momentarily lowers the mip level. If the mip remains high, it indicates to the inquiring DE that some other DE is also trying to transmit and the bus race has been lost. The DE that just looked must now go back and wait for the new transmission to be completed. Clearly this system assigns an implicit priority in that the longer the assigned wait period, the higher the priority.

This structure can be used either for single word or block data transmissions. If the mip line is lowered after each word of block transmission, time response to critical data is improved, but bus transfer efficiency is reduced by the waiting period to resolve the bus race for each word. Also, each word must contain both destination address and data identifier in this case. If the mip line is only lowered

at the end of a block data transmission, time critical data response may not be adequate, since the subsystems operate asynchronously.

Another potential problem is the implicit priority assignment, which must be made on a configuration-wide basis. The digital processor can output priority data for each DE as part of the prelaunch preparation of the weapon. However, some subsystems output more than one type of data, and the priority of each type may vary. This structure can theoretically accommodate varying priorities for a single DE, but the control structure will be complicated.

The central control structure usually uses an auxiliary communication structure for communication of each DE with the bus controller, as shown in Figure 39. When a DE wishes to use the bus, it informs the bus controller which allocates the bus on the basis of the system state which includes all current and pending requests from DEs. The requests from the DEs to the controller furnish data available/data required types of information in accordance to the asynchronous operations of the weapon subsystems. This function was performed by the interrupt inputs of the three point designs. Alternatively, the central bus controller may operate without the auxiliary structure by using a polling procedure. This procedure would require that the controller sequentially request status (interrupt) information via the data bus from each subsystem and perform a bus allocation on this basis. The bus response to interrupts with this form of central control is equivalent to that of the rotating window control structure with a window time corresponding to two message word times (request, response) for each subsystem in the weapon. An extremely high bus capacity would be required to meet the response time requirements for time critical data transfers, especially since the polling procedure must be performed at a fixed rate independent of data and control transfers occupying the bus.

The obvious alternative to a polling procedure is the addition of a second bus to allow asynchronous interrupt transfers between the weapon subsystems and the central controller. The requirement for asynchronous operation of this interrupt bus dictates a distributed control philosophy. Of the distributed control structures, the bus race

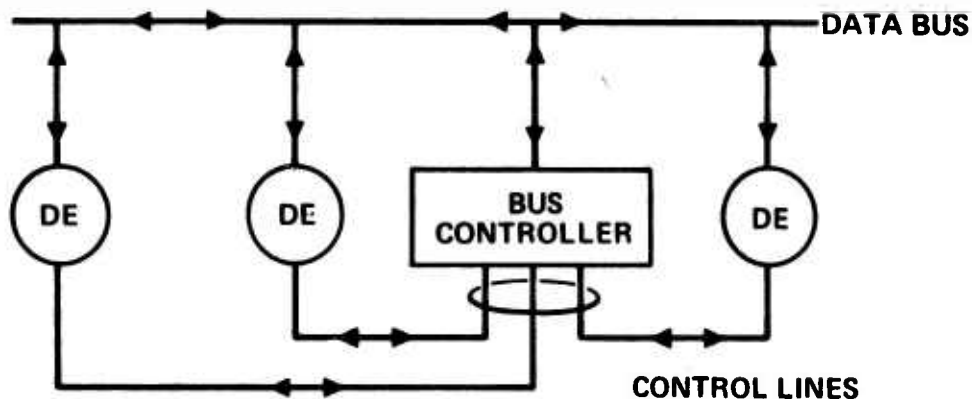


Figure 39. Central Control Structure

control provides the best time response. Since interrupt data can be transferred in a single word, and the interrupt traffic is relatively low, the problems associated with bus race control of the data bus are essentially eliminated.

The use of a separate interrupt bus to transfer data requests between the weapon subsystems and the central data bus controller is considered a necessary adjunct to central control. In order for the central controller to perform its function, it must send control words to the source and destination subsystems. These words both set the subsystem data transfer mode (transmit, receive) and identify the data being transferred. Central bus controller hardware complexity is minimized by performing the dynamic allocation in the DP software. The interpretation of interrupt signals is an integral part of this function.

#### 7.1.2 Bus Message Word Length

The bus message word length is determined both by the information which must be transferred in each message word and the coding technique used in the word. Generically, the information content of the message word can be divided into three fields: data, identification, and error detection. The data field contains information in a form useful within the destination subsystem and includes data, subsystem status and control parameters, and coded interrupt commands. A data field of 16 bits has been used for compatibility with DP word size. The identification category includes source and destination subsystem information as required by the bus control elements and indicates the purpose of the word data content within the destination subsystem. The error detection category contains one or more parity bits as required to meet the transfer reliability specification. The coding problem is associated with the identification category in which maximum information capacity is desired to insure compatibility with total weapon system requirements. Conversely, the coding should allow minimum complexity in the interface circuitry between the bus and each subsystem. In the following discussion, this interface circuitry is designated as a bus interface unit (BIU) and includes all necessary bus control elements.

#### Message Identification Coding

Three coding techniques for the message identification information are shown in Figure 40. The weapon configuration information required in the combination of BIU and subsystem varies with coding technique. If a system-level identifier is used to define the data content of each message word (or block of words for block transmission formats), no weapon configuration information is required in either the BIU or subsystem. System level identification implies that any source subsystem may output each of its message words with an identifier which is only a function of the data content and completely independent of the destination. Furthermore, the destination subsystem operations on the message data content are completely determined by the identifier. If the operations of the destination subsystem depend on which of the alternative subsystem sources of a specific parameter actually furnished the

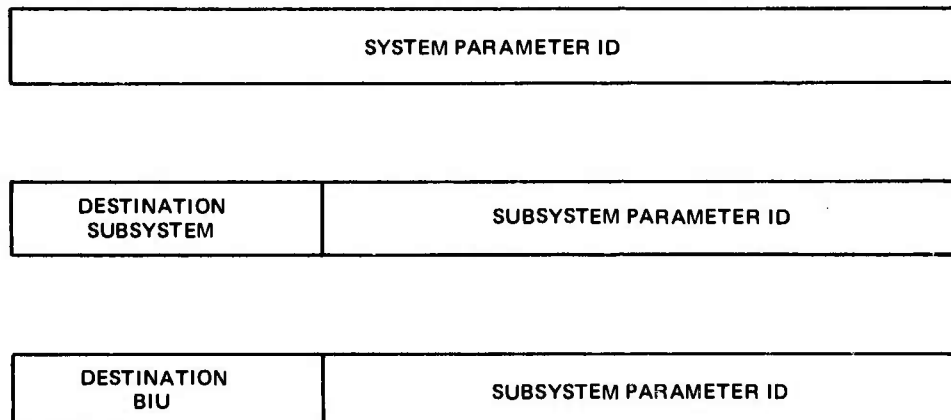


Figure 40. Message Identification Coding

data, a different identifier must be assigned to the parameter for each source. This implies that the identifier may be divided into two fields: source identification, and parameter identification. In the most general case, this format is required and results in a large number of bits in the message to allow for specification of all possible types of data, control signals, and interrupts on a system wide basis. Each subsystem must compare the identifier of each message word on the bus to all identifiers pertinent to the subsystem. When a word is accepted, the identifier must be interpreted by the subsystem-- this mapping from message identifier space to subsystem operation space is non-trivial. The subsystem must also store identifiers for each of its pertinent bus message outputs. This identification coding technique can involve considerable subsystem complexity and should only be considered for distributed bus control techniques.

The second and third coding techniques both use two fields to facilitate the decision process in the combination of BIU and subsystem. Either technique may be used for both central and distributed control philosophy. The first field of both techniques identifies the destination of the message, and implies that weapon configuration information must be contained in the bus control element which causes the message word to be sent. The only difference between the two techniques is the coding of the destination: subsystem versus BIU. Since only the DP has complete weapon configuration information, a distributed control philosophy would require that the DP furnish destination information to each message source for each type of message output in the most general case. However, in any weapon configuration, all bus communication is between the DP and the other weapon subsystems. Direct communication between the other subsystems is generally not required, and, if desired, usually involves conditioning of the parameters which is a DP function. Thus, any distributed element need only insert a destination identifier corresponding to the DP if distributed bus control is used. The DP contains all configuration information required to determine the appropriate destination for any message which it originates.



The tradeoff to determine the coding of the destination identifier favors the BIU number rather than subsystem number for the following reasons. If a number is assigned to each BIU independent of the subsystem to which it is connected, the destination ID field need only account for the maximum number of subsystems in any single weapon configuration (estimated as 16) rather than the total number of subsystems in the weapon system. In order for the DP to identify the weapon configuration, it must access the subsystem identifiers. This process is simplified by the BIU coding, since the subsystem identifier can be an addressable output of each subsystem via its BIU and the DP need only access all BIUs to obtain the subsystem identifiers.

The coding of the parameter ID field must allow all parameters input to the DP in any weapon configuration to be uniquely identified, since the number of parameters between the DP and any single weapon subsystem is small. In lieu of making a detailed study of all possible coding schemes for parameter identification, a 12 bit field corresponding to the maximum expected DP read/write operand memory space has been assumed. This allows simple allocation of blocks of memory for alternative subsystems of each type for the distributed bus control techniques. Each subsystem output parameter can then be assigned a fixed address which is stored within the subsystem and accessed by the bus controller as part of the message word. A smaller address space is allowable for central bus control, since the DP can dynamically allocate its memory for each configuration. However, all message words on the data bus must be the same length for simplicity of implementation in the bus controller.

#### Bus Message Formats

The bus message word contents applicable to the various bus topologies for both single word and block data transmissions are shown in Figure 41. The number of bits shown for each field are based on the coding philosophy presented in the previous paragraph. Small variations in these numbers will not have any significant effect on the bus design tradeoffs. The ordering of the information within the word is arbitrary and may be adjusted for minimum hardware implementation.

Analysis of these word formats and bus operating procedures results in the bus transfer efficiencies shown in Table 14. These results are normalized to an N word block data transfer since the majority of the data sources provide more than one word at their transfer rate (see Table 12). The serial structures obviously are more efficient than the ring structure and the block format for the serial structures is more efficient than the single word format.

As discussed previously, interrupts must be transferred to synchronize the subsystem operations. The serial/bus race design can also transfer coded interrupt data at a cost of 40 bits for each command. A separate interrupt bus is used for the serial/central control design and the interrupt word format is shown in Figure 42 (13 bits).



TABLE 14. BUS TRANSFER EFFICIENCY

BUS TOPOLOGY/CONTROL	TOTAL BITS FOR 16N DATA BITS
RING	80N
SERIAL/BUS RANGE	
SINGLE WORD FORMAT	34N
BLOCK FORMAT	20 + 20N
SERIAL/CENTRAL	
BLOCK FORMAT	40 + 20N

RING STRUCTURE

P (1)	OP CODE (3)	SOURCE (4)	DESTINATION (4)	DATA ADDRESS (12)	DATA (16)
----------	----------------	---------------	--------------------	----------------------	--------------

SERIAL STRUCTURE

SINGLE WORD, BUS RACE

P (1)	OP CODE (1)	DESTINATION (4)	DATA ADDRESS (12)	DATA (16)
----------	----------------	--------------------	----------------------	--------------

BLOCK TRANSFER

CONTROL	P (1)	OP CODE (3)	DESTINATION (4)	DATA ADDRESS (12)	1/ BLOCK:BUS RACE 2/ BLOCK:CENTRAL
---------	----------	----------------	--------------------	----------------------	---------------------------------------

DATA	P (1)	OP CODE (3)	DATA (16)
------	----------	----------------	--------------

Figure 41. Bus Message Formats

P (1)	DESTINATION (4)	INTERRUPT ID (8)
----------	--------------------	---------------------

Figure 42. Interrupt Bus Word Format

### 7.1.3 Weapon Bus Tradeoff Summary

The ring structure is not recommended because of long transfer times for critical data and low bus transfer efficiency. The serial/bus race design with block transfer format has the highest transfer efficiency, but potentially cannot meet the time requirements for critical data since a low priority block transfer from one subsystem cannot be interrupted for a high priority transfer for a different subsystem.

Only the serial/central control design provides a system-wide bus allocation capability, but a separate interrupt bus is required. The serial bus structure with the central controller installed in the digital processor is the recommended design for the data bus. This design provides good transfer efficiency for data blocks, and the system state data required for bus allocation is contained in the digital processor software. The central controller hardware complexity is minimized by making the allocation decisions in the software. This allows a hardware implementation of the central controller which is independent of weapon configuration.

Central control of the interrupt bus is not recommended since interrupt generation by the weapon subsystems is asynchronous. Since interrupt data can be transferred in a single word, bus race control of the interrupt bus is recommended. Priority can be set for each subsystem via the data bus as part of the prelaunch weapon preparation on the basis of interrupt rate. This technique provides short wait times for critical interrupt transfers.

## 7.2 PROCESSOR SYSTEM ARCHITECTURE

In the preceding subsection, a weapon bus configuration was chosen for the modular weapon. In this subsection another aspect of system architecture is examined, i.e., processor system architecture. The requirements for the digital processor are based upon the CORE function: System Management, Flight Control and Strapdown Inertial Reference Function. The question of interest here is, should these functions all be performed in a single processor or should they be distributed among two or more processors.

In the analysis reported below, it has been assumed that the system management function would not be split up. That is, basic integration and control function should reside in a single processor. Therefore, the tradeoff areas are concerned with performing the flight control function and the strapdown inertial reference function in separate processors. It was not the purpose of the analysis to create new subsystem designs; the analysis is based on existing designs. Specifically, the flight control subsystem used in the analysis is based on the GBU-15 flight control.

The processor system architecture analysis is concerned with the effect on total processing requirements of distributing the CORE functions among an arbitrary number of processing elements. For this

study, a distributed architecture is defined as a number of processing elements interconnected via the weapon bus. These processing elements may be collocated with other weapon subsystems and provide an adapter module function for a subsystem, but are not necessarily dedicated to the processing functions associated with the subsystem. Any digital processing element physically located within a weapon subsystem and totally dedicated to functions of the subsystem is not part of the digital processor architecture. The central processor architecture is a special case within this definition of a distributed processing architecture.

In order to perform any function in any of the digital processing elements, the generic procedure shown in Figure 43 is followed. Some of the steps shown are trivial when the input data source and output data sink are within one digital processing element. If the data source and/or sink are in different physical locations, the data transfers are weapon bus operations. The steps shown in this figure do not include the weapon integration process in which the required processing functions and data sources and sinks are identified for the weapon configuration. The executive software operations discussed in the following section are required to energize the function processing and interface with the weapon bus, and are implicit requirements.

The entire System Management function should be performed in a single processing element which also includes the weapon data bus controller. This processing element is designated as the control processor (CP) since it not only controls the other elements of the distributed processor, but also the other weapon subsystems. The CP must contain the complete executive software structure described in the following section to support the integration functions. The degree to which this executive software must be duplicated in the other distributed processing elements depends on the partitioning of the remaining functions.

#### 7.2.1 Flight Control Function in Separate Processor

From previous experience with the GBU-15 autopilot, it is known that there are two factors which determine processor throughput requirements. These are the computational task itself and the propagation delay requirements to preserve stability. Of particular interest in the analysis is the effect of the weapon bus on the propagation delay requirement (the GBU-15 does not have a multiplexed bus).

The processor which performs the flight control function is designated as a flight control processor (FCP). The operations required by the CP to support the FCP are dependent on both the technique used to interconnect the flight control sensors and actuators with the FCP (weapon bus versus direct I/O) and the method of real time control of the FCP functions. These factors are not independent and are discussed in the following paragraphs for the configuration examples shown in Figure 44. The only assumption made on the physical location of the elements shown in the figure is that direct I/O connection is only allowed if the elements are located in the same weapon section.

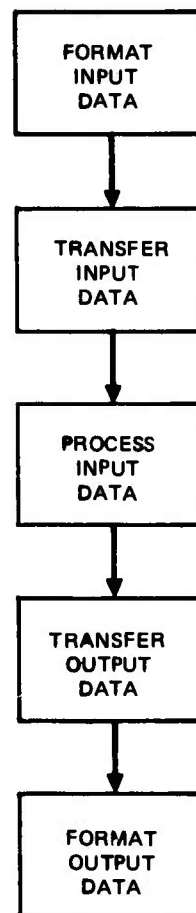


Figure 43. Processing Sequence for Digital Functions

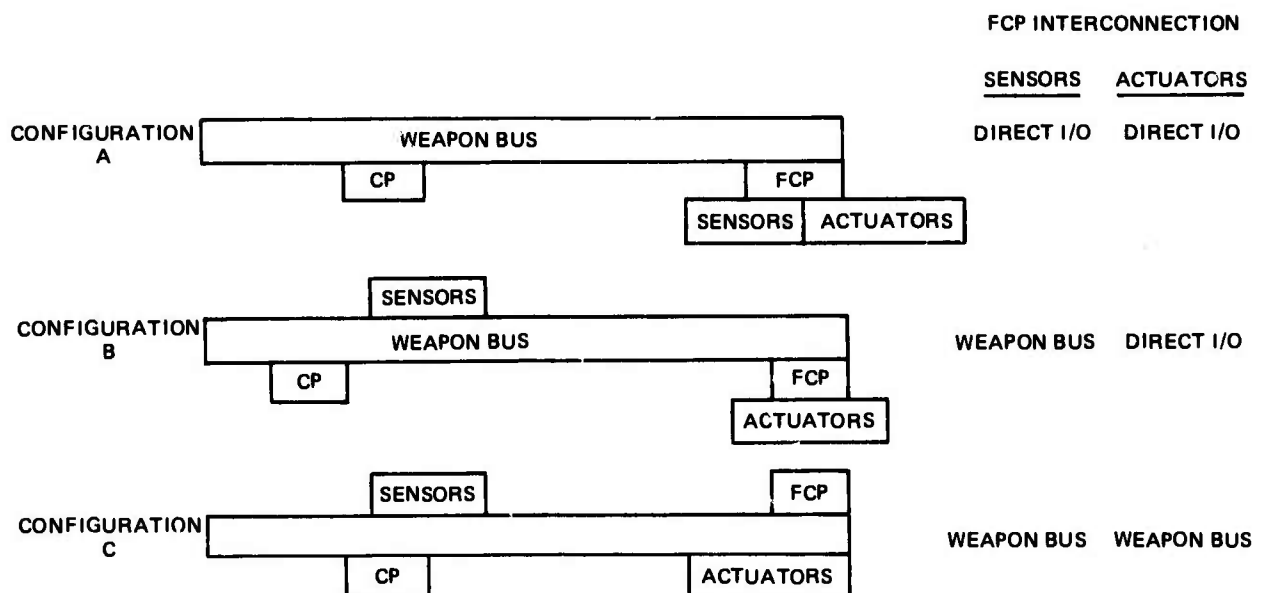


Figure 44. Distributed Processing Configuration Examples - Flight Control Functions

The following assumptions have been made concerning the flight control function:

1. Both the stabilization and steering functions are performed at a fixed iteration rate. If the iteration rates vary with airframe, the CP determines the appropriate rate for each weapon configuration.
2. The inertial sensor output data format may be either digital or analog. In either case, a sample command at the appropriate iteration rate is required to initiate the formatting of the sensor data for use in the FCP. This command is generated by the real time control element (CP or FCP) and the response time of sensor and formatting circuitry may vary with sensor subsystem. Sampling and format conversion is a function of the sensor subsystem.
3. The actuator subsystem may be analog requiring format conversion of the FCP digital output data. This conversion, if required, is performed by the actuator subsystem.

The sequence of operations involved in each of the flight control functions (stabilization and steering) are shown functionally in Figure 45. These operation sequences must be partitioned between the CP and FCP and the performance requirements on the elements of the distributed processing system (CP, FCP, and weapon bus) must be determined for each of the configuration examples.

The principal processing element requirements of interest are throughput rate and memory capacity. The interaction of processor throughput and weapon bus capacity must also be determined. The following assumptions have been made in the study:

1. The CP and FCP have identical processing capability (throughput, instruction set).
2. The CP and FCP both contain the executive software, with the exception of the bus control function which is only installed in the CP.
3. All system level timing is controlled by the CP. This implies that appropriate interrupts are sent from the CP to the other elements to synchronize their operations.
4. The stabilization function propagation delay requirement is 600 microseconds (GEU-15 requirement) to complete all operations from sensor output sampling through conversion of the actuator commands.
5. Sensor output data formatting requires 20 microseconds per data word (set by analog-digital conversion).
6. Sensor data transfers to the FCP will be in block format for Configurations B and C under control of the CP in response to an interrupt from the sensors when all data has been formatted.

7. Digital-analog conversion of the actuator commands requires 5 microseconds per data word.

8. The system control procedure used for the analysis is based upon the bus control philosophy developed in the preceding section.

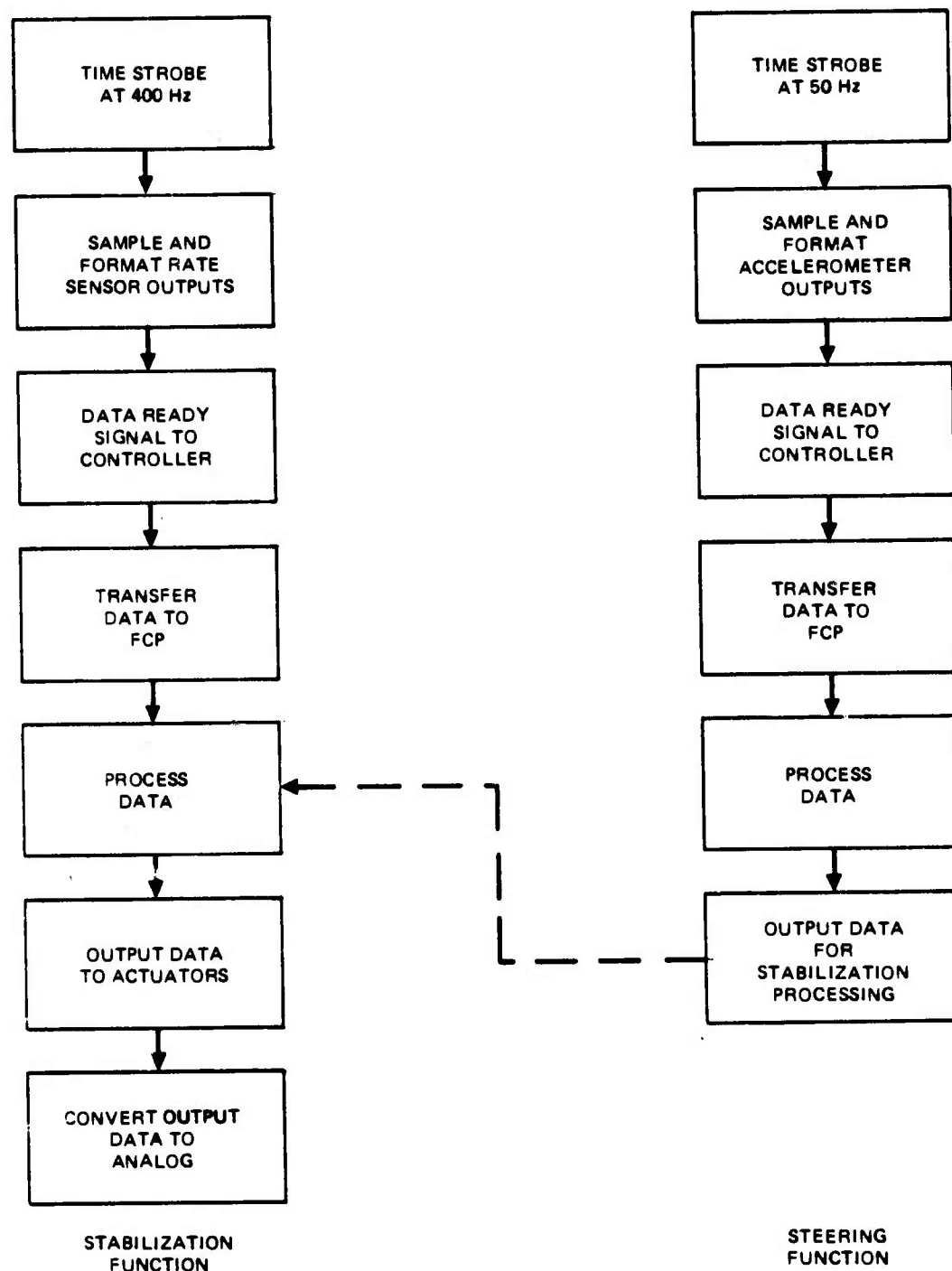


Figure 45. Flight Control Function Sequences

The instructions executed for each software control function are shown in Table 15. These are in accordance with the system control procedure. The processing system operations required for the stabilization function are shown in Figure 46 for each of the three distributed processing configurations. The total number of instructions executed and weapon bus words (data and interrupts) transferred were derived and are shown in Table 16. This table also shows the equivalent numbers for performing all operations in a single processor. For the central processor case, the processor is located at the position the FCP takes in Figure 44. BC is the configuration derived from configuration B by placing the single processor with the actuators. CC is the configuration derived from configuration C by deleting either one of the two processors shown and including all functions in the remaining processor.

The total time allowed to perform the stabilization function (from sampling to delivery of output data to the actuators) is 600 microseconds. Sixty microseconds are required for sampling and analog-to-digital conversion (20 microseconds per word). Another 15 microseconds are required for digital-to-analog conversion at the actuators. This leaves 525 microseconds for the required weapon bus transfers and the processor tasks. The processor throughput capability is a function of the bus transmission rate; the slower the bus transmission is, the

TABLE 15. INSTRUCTIONS EXECUTED/ITERATION


FUNCTION	SHORT	LONG
INTERRUPT SERVICE	25	
TASK SUPERVISOR	75	
 (INTERRUPT OUTPUT FROM CP)	3	
DATA TRANSFER	30	
STABILIZATION COMPUTATIONS	350	20

TABLE 16. SYSTEM OPERATIONS REQUIRED FOR EACH STABILIZATION COMPUTATION

CONFIGURATION	INSTRUCTIONS		
	SHORT	LONG	WEAPON BUS WORDS
A	350	20	0
B	683	20	7
BC	580	20	6
C	816	20	13
CC	610	20	9



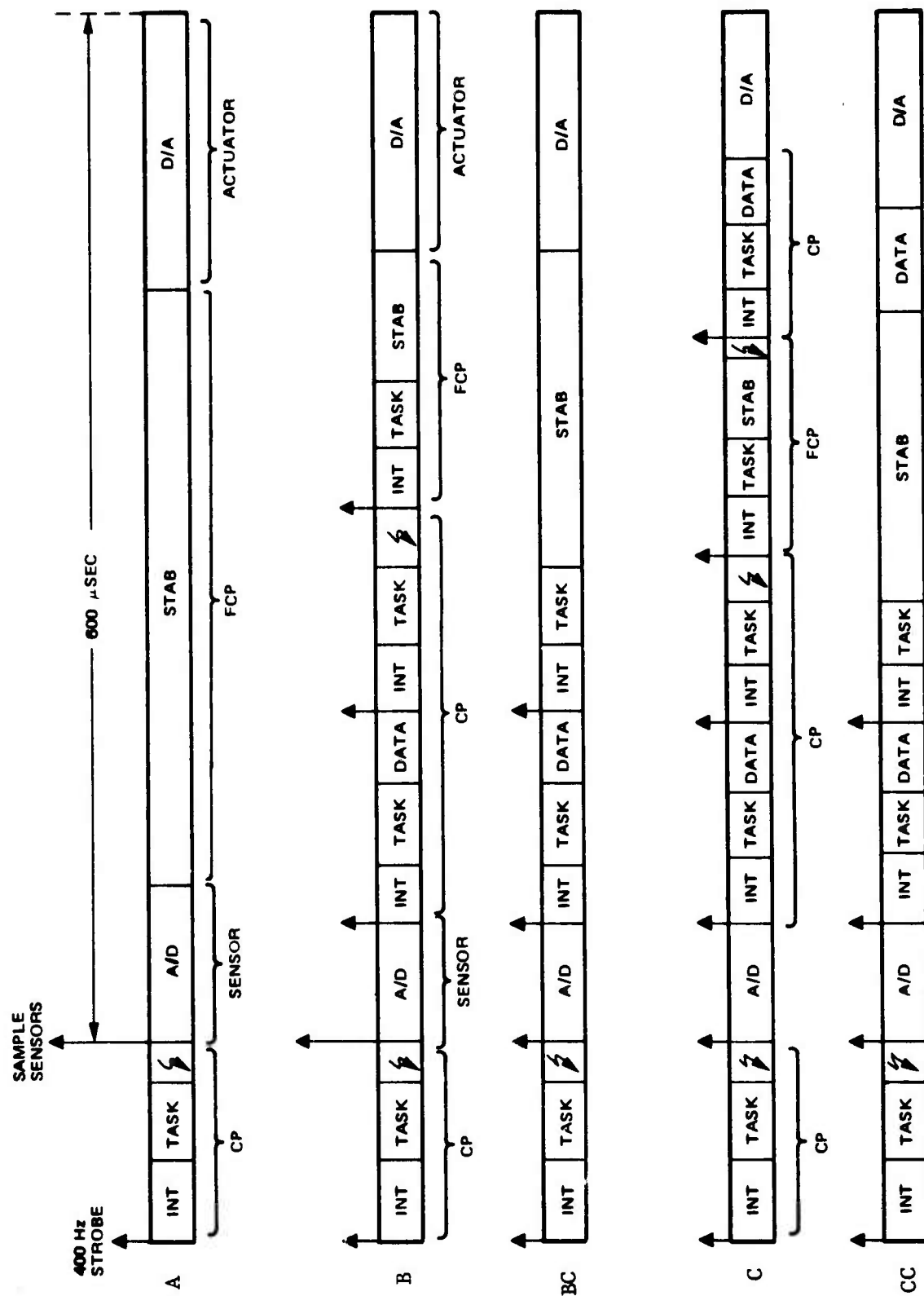


Figure 46. Stabilization Function Operations

faster the processor must operate. For each configuration, the processors must perform the instructions shown in Table 16. In the distributed processor case, the FCP performs 350 short instructions and 20 long instructions for the stabilization function. The other instructions are communication control or interrupt responses and are shared by the two processors. It must be remembered that the two processors operate in series. Note that there are different numbers of words to be transferred on the bus for the different configurations.

The tradeoff of bus rate versus required processor throughput capability is shown in Figure 47. For this tradeoff it was assumed that long instructions were performed at the same rate as short instructions. It is noted that for the distributed processor cases, each processor must have the throughput capability shown.

Configuration A has the lowest throughput requirement per processor. In this configuration no bus transmissions are required. It is equivalent to the GBU-15 situation. This configuration, of course, does not fit in with the modularity concept described in Section VI and so is not an allowed configuration. If it were allowed, however, the CP

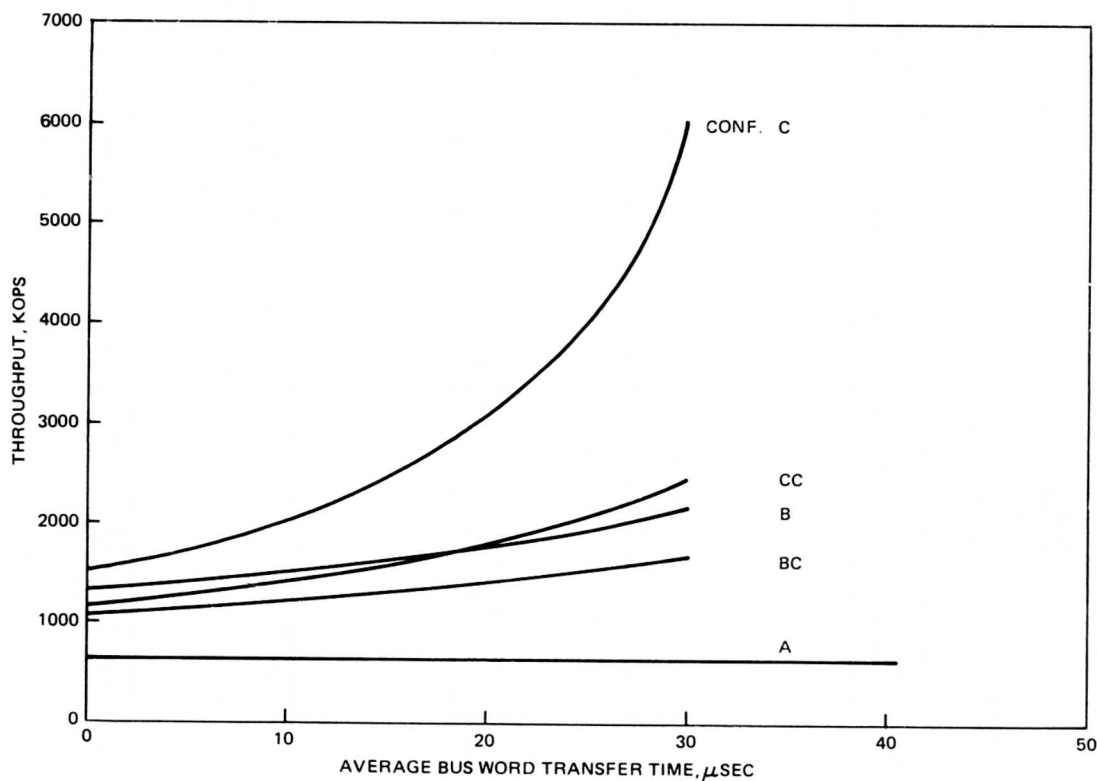


Figure 47. Throughput versus Bus Rate

should also be placed in the same spot as the FCP and the functions should be combined in a single processor. (The results of the cost analysis reported in Section IX supports this arrangement.)

Of the remaining configurations, C has by far the highest throughput requirements and is clearly not a good choice. The three remaining configurations have about the same throughput requirements per processor. However, B requires two processors while CC and BC require only one. There would seem to be no advantage to configuration B.

Configuration CC best meets the modularity requirements. (It does not require any special positioning relative to subsystems.) It is the recommended configuration.

The above results are consequent to the necessity to transmit data over a multiplexed bus. It is possible that other bus control procedures would modify the above results (alternate procedures were not investigated in this context). However, no matter what the procedure used, the throughput requirements for the processors will always be higher when the bus must be used (configurations B, BC, C and CC) than when it is not used (configuration A).

It is possible that some control procedures will reduce the peak processor requirements described above for the distributed case. What is needed is a procedure that treats the FCP as a special case, so that the operations of the CP are not in series with the operations of the FCP. This requires that the CP dedicate some of its time in a wait mode (and also reserve the bus) so that it can overlap at least some of the system control process with the stabilization computations. Such a procedure will lower the peak throughput requirements on each processor, but not by as much as a factor of two. Even if the requirements were lowered by a factor of two, there is no cost advantage. In fact, the cost study reported in Section IX indicates that the cost will be greater for the two distributed processors as compared to the single processor with twice the throughput.

A strong argument against the use of such control procedures is that it violates the modularity concept. If there is no economic benefit, then it should be eliminated in favor of the more general procedure used in the above analysis.

One possible configuration not treated in the above analysis is that in which two or more processors are collocated. This kind of arrangement can reduce the required number of bus transfers. In fact, the number of bus transfers can become equal to that required for the single processor case. In order to take advantage of this arrangement there would have to be a very efficient direct I/O connection between the processors, such as a shared memory. Furthermore, the tasks would have to be divided up differently than in the cases analyzed above. In particular, the flight control function could be divided up. If one processor performed the system management and the roll channel and the other processor the lateral channels, there would be a slight reduction in the requirements for each processor. But two processors are

used in this configuration. The results of the cost analysis (Section IX) apply here. The results of that study indicate that dividing a processing load between two separate processors in general leads to increased costs relative to a single processor, unless the single processor implementation pushes the state of the art. The exception is found when the two processors share a large amount of resources (same box, shared memory, shared I/O). This latter case is the multiprocessor and is not really a distributed configuration. It is just another way to configure the single processor.

In summary, separating the system management function and the flight control function and performing them in different processors leads to an increase in system cost for the modular weapon. Each processor must have the same (or very close to the same) throughput capability as a single one would have. This result is a result of the propagation delay requirement for the autopilot and the fact that data is transmitted on a time multiplexed bus rather than a dedicated harness. The propagation delay requirement for the autopilot places a peak throughput requirement on the processing system. As will be seen later, this is the requirement which really sizes the processor.

#### 7.2.2 CORE Function Processing Requirements

The analysis of the separation of the flight control function from the control processor concerned itself primarily with the requirement on peak processor throughput to accommodate the stabilization function. There is no comparable concern in the case of the strapdown inertial reference function. Therefore, before investigating the separation of that function from the CP, it is helpful to review the total processing requirements for the CORE functions to give a basis for analysis.

The processing requirements are given in Table 17. The system management function includes a communication control based on the bus philosophy described in subsection 7.1. This is the same control process as used in the above analysis. All external subsystems data sources and sinks are assumed to use the weapon bus for communication with the processor.

The requirements for system management also involve the assessment of subsystem status data to accomplish real time control of the weapon subsystems, including the digital processor and weapon bus. The throughput requirement is based on the requirements for the weapon control unit (under development for the GBU-15 weapon system) to perform the equivalent function. (This is the logic function of the WCU.) Provisions have been made for the intersubsystem communications required for status word inputs and outputs. The system management function energizes self-test in the other weapon subsystems and in the applicable digital processor software functions in response to status inputs from the AGE. The memory requirements for the executive software are included in the system management software.

TABLE 17. CORE FUNCTION PROCESSING REQUIREMENTS

FUNCTION	PROGRAM SIZE	OPFRAND MEMORY	AVERAGE THROUGHPUT KOPS			BUS RATE (words/sec)	
			TOTAL	SHORT	LONG	DATA	INTERRUPT
FLIGHT CONTROL	600*	240*	172	162	10	3600	900
STRAPDOWN INERTIAL REFERENCE	1700*	540*	103.2	91	12.2	840	225
SYSTEM MANAGEMENT	700*	500*	407	407	0	2400	**
TOTAL	3000*	1280*	682.2	660	22.2	6840	1125
*MEMORY REQUIREMENTS GIVEN FOR A SINGLE CONFIGURATION ONLY.							
**INCLUDED IN ABOVE ENTRIES							

The throughput requirements shown are average. It should be noted that the average requirements are considerably less than the peak throughput requirements indicated in Figure 47. In that figure, a throughput of the order of 1.5 million operations per second is indicated, and this is with the assumption that long instructions are executed at the same rate that short instructions are.

Also to be noted is the fact that the memory requirements shown in the table are sufficient for only one configuration. The complete modular weapon, with several configurations, will require more memory.

### 7.2.3 Strapdown Inertial Reference in Separate Processor

As shown in Table 17, the strapdown calculations, in the average sense, fit easily into the peak throughput requirement established by the flight control stabilization function. The strapdown calculations do have a time requirement on them. The processing functions associated with the strapdown inertial reference itself must only be completed within the iteration period of each function. However, the alignment/correction filter computations must be completed in less than 100 milliseconds. This propagation delay effect requires a throughput capability of about 200 KOPS. Since it can be interrupted by the flight control computation, it does not add to the peak throughput requirement. Thus, the strapdown processing can be included in the CORE function processor without adding to the throughput requirements.

The strapdown inertial reference computations could be performed in a distributed processing element with relatively low capability (~250 KOPS throughput). However, the required capability of the control processor would not change, and system cost would increase due to the added processor. Since there is no system advantage to putting this function in a separate processor, and there is a cost penalty, it is not a recommended approach.

#### 7.2.4 Summary

In the context of the modular weapon, with a time multiplexed weapon bus, there is no system advantage in separating either the flight control function or the strapdown inertial reference function and performing these functions in separate processors. There is a cost penalty if one does separate them.

### 7.3 SOFTWARE DESIGN

The digital processor software is an integral part of the modular weapon system and must support the modularity of the system. This software plays a key role in the functional integration of the weapon subsystems as well as performing many of the computational and decision operations for individual weapon subsystems.

The digital processor functions can be conveniently divided into two categories. One is a group of functions that are common to all configurations and are called the CORE functions. These functions will be performed in all weapon configurations regardless of what subsystems are included in the weapon, but there will be variations within these functions corresponding to differences between weapon configurations. The second category is called configuration dependent functions. These are functions that are performed only when a particular subsystem is installed in the weapon, and only if the processing function is appropriate for performing in the digital processor.

The CORE functions are flight control, strapdown inertial navigation, and system management which includes configuration identification, communication control, sequencing, initialization, and self test. The CORE functions, although they will be present in every configuration, will have to be able to adapt to the differences among various configurations. The flight control function, for example, will have to be able to adapt to the different aerodynamics and terminal trajectories of the various configurations. The system management function in particular will have to be able to identify the present configuration and alter the sequencing and communication control sub functions to fit the hardware modules that are actually installed in the weapon. It will also have to vary the self test procedure to test only the functions and systems used in the configuration.

The configuration dependent functions are associated with specific midcourse and terminal guidance subsystems which may be installed in the weapon. Examples of subsystem signal and data processing functions were defined in the point designs of Section V. Although no specific recommendation has been made concerning configuration dependent signal and data processing functions to be performed by the digital processor, the factors in the decision process are discussed in subsection 7.6. The software structure must be capable of supporting both the CORE functions and desired functions of this type. Many of these functions may be designed into the software, but in any given configuration only a few, if any, of these functions will be performed. It is the



responsibility of one of the CORE functions, system management, to determine what subsystems are installed and which configuration-dependent functions are to be performed, if any, and to provide sequencing and communication control as appropriate.

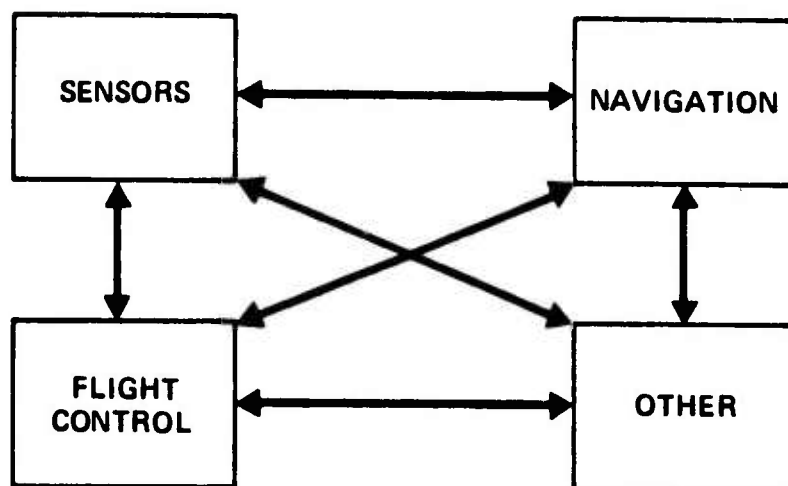
Many of these functions include operations which must be performed within a limited amount of time after the occurrence of some event external to the digital processor or some time event from the system clock. This means that the execution of many of the processor functions must be initiated by interrupts received by the processor. Furthermore, the particular function performed when an interrupt is received can vary with the subsystems installed and with the phase of flight. Therefore, the software must include a flexible interrupt structure which can be modified by the system management function. In addition, the software must be able to grow in the future by adding new software modules without altering the basic structure of the software and without altering many existing software modules.

One program structure (shown in Figure 48) that has been used successfully in many small programs directly interfaces each software function with every other software function with which it must communicate and with any external subsystems. This structure is adequate for small programs and can be quite efficient because there is little overhead required. However, it suffers from a large number of interfaces which can grow as the square of the number of functions making expansion of the system difficult. The large number of interfaces also makes modifications of any of the functions difficult because the modifications tend to propagate across the interfaces into other functions. In the past this type of structure has contributed to the high cost of software development and maintenance.

A more suitable structure for the DP is one that utilizes an executive. An executive is a collection of supervisory functions that manage the resources of the processor. All interfaces between software functions or between software functions and external subsystems are through the executive. With this structure (shown in Figure 49) the number of interfaces only goes up linearly with the number of functions. Changes in one software function do not propagate into other software functions very often because of the isolation provided by the executive. Also, it is a relatively simple matter to add more functions to the system. The processor functions are partitioned into software modules that are largely independent of each other except for the common data that they operate on. This structure facilitates the use of a top-down design methodology for developing the software.

The direct interconnection method is somewhat faster when a small number of programs must be controlled, however, when more programs are to be controlled, it requires either a large number of priority levels or some software routine to sort out the priority levels. It also results in a less general structure which is harder to modify or add to. Because of the desire for modularity and the number of programs, the use of a simple real time executive is chosen for this system.





**DIRECT INTERCONNECTION OF PROGRAM MODULES  
RESULTS IN A LARGE NUMBER OF INTERFACES AND  
EXCESSIVE INTERDEPENDENCE AMONG MODULES**

Figure 48. Direct Interconnection of Program Modules

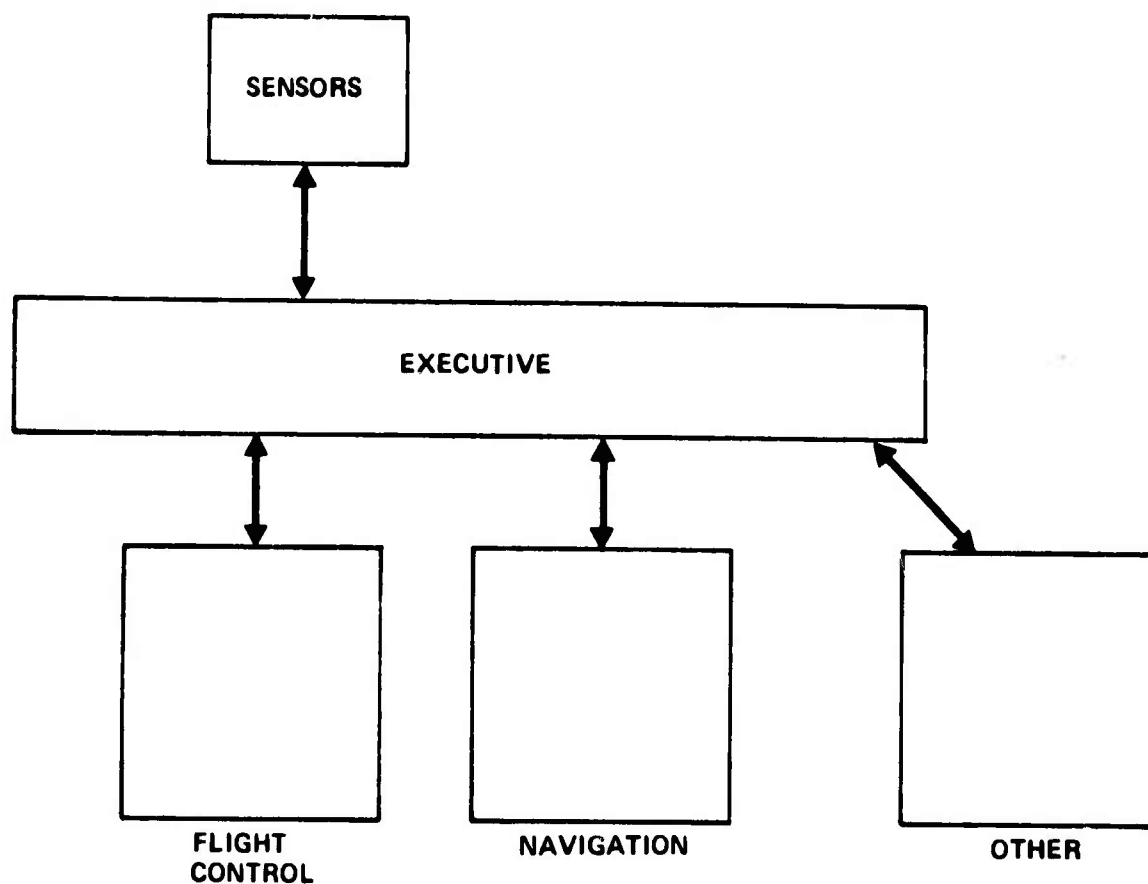


Figure 49. Program Structure Using an Executive

The executive is an organized collection of software routines (shown in Figure 50) designed to manage the processor resources. It is responsible for interfacing with all interrupt hardware and input/output equipment and for supervising the execution of all software modules. The executive acts as a buffer between software modules and the hardware by performing all input/output operations thereby making software modules independent of the mechanization of input/output hardware. It also acts as a buffer between software modules, providing a common way of interfacing one module to another. The executive is a permanent part of the system that is common to all configurations.

The processor functions are partitioned into units called tasks. Each task is a unit of work that is to be performed as a result of some external event or time event or command from another task. Tasks can range in size from a few instructions to a few hundred instructions, and in execution time from microseconds to seconds. Some tasks may only be executed once while others may be executed from 100 to 400 times per second. Tasks are characterized by a unique identification number, a starting address, and a priority. Tasks are called by each other or associated with external events only by their ID number. Only the executive needs to know the location and priority of a task and since this information is contained in a table, it can vary from one configuration to another without the tasks having to be changed.

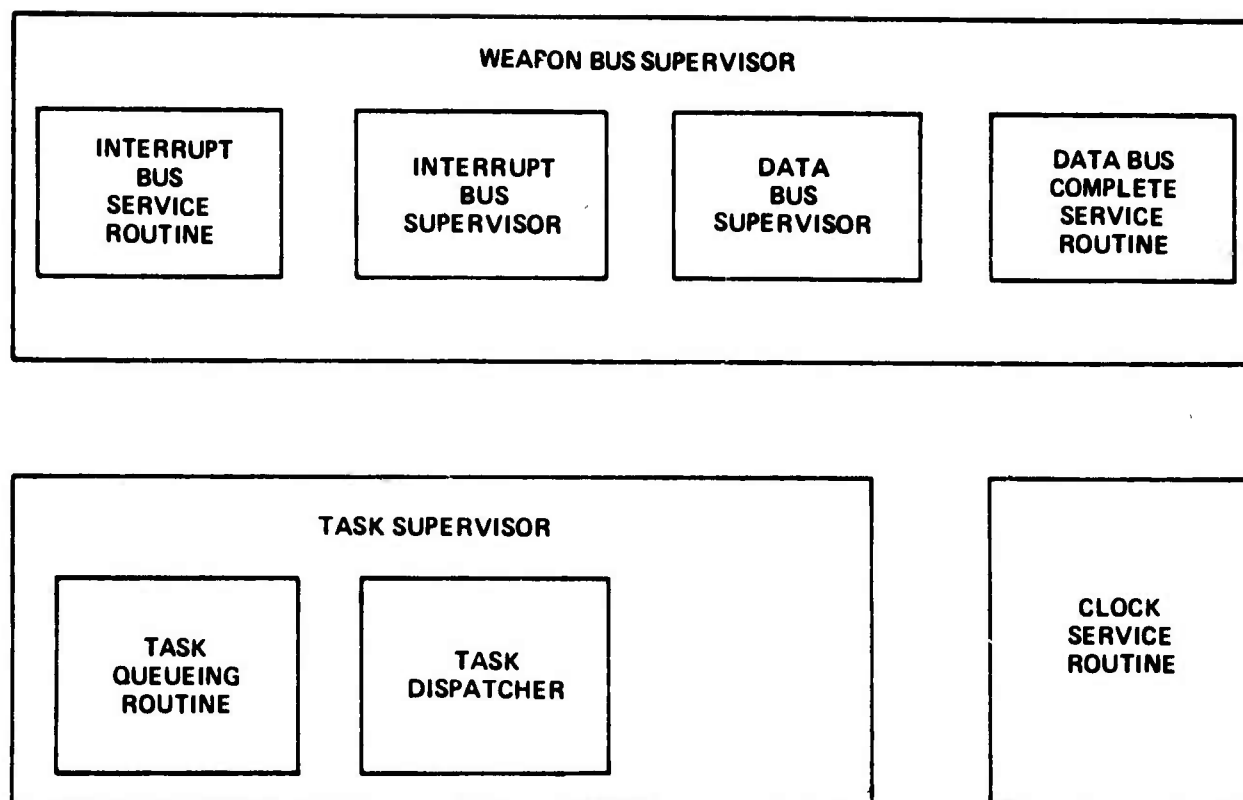


Figure 50. Executive Routines

In determining how to partition a processor function into tasks some general rules should be followed.

1. Each task should consist of a single operation at some level of abstraction in the description of the function. At a lower level of abstraction the task might include many operations but there should be some level at which the programmer can consider the task as a single operation. Otherwise it should be broken down into smaller tasks.

2. A task should not have delay loops longer than 30  $\mu$  seconds in it that wait for some external event to be completed. If a task must initiate an external event, such as the transmission of a block of data from a weapon subsystem to the DP, and has no other operations to perform until the data block has been received by the DP, then it should be broken into two tasks. The first task would initiate the external event and then end by returning to the executive. The second task would execute when the data transfer is complete. During the time in between the executive can start executing some other task so the time is not wasted.

3. A task should not have a small subset of operations that have a much higher priority than the bulk of the task. Instead it should be partitioned into two tasks which have different priority levels. One task containing the bulk of the operations would run at a low priority, and the remaining few operations would be performed in a separate task at a higher priority.

Whenever two or more tasks operate on the same data, whether two tasks are operating on common data from an external source or a task is operating on data produced by another task, the common data will be converted to a format and scale factor and stored in locations decided upon at the system level. This will prevent incompatibilities in data formats and scaling that could otherwise occur.

All these tasks must be tied together and executed in a sequence appropriate to the weapon configuration. This will be accomplished by two levels of software. One of these levels is the executive which causes software modules to be executed in response to signals from external devices or in response to commands from other software modules. The executive does this without any knowledge of the weapon configuration or what each module is supposed to do. This knowledge is contained in the system management module which is the second level of software that controls the sequence of execution of all tasks.

The system management module is the glue that holds all the other tasks together to form a working system. This module must interrogate all the external devices that may be connected to the processor to determine what configuration weapon it is in. After determining the weapon configuration it must set up linkages that will cause the proper tasks to be executed to process the data available from the installed hardware modules and to provide the proper outputs to control the weapon. This is accomplished by setting up tables that link the occurrence of an event to the execution of a task. These tables (shown in Figure 51)

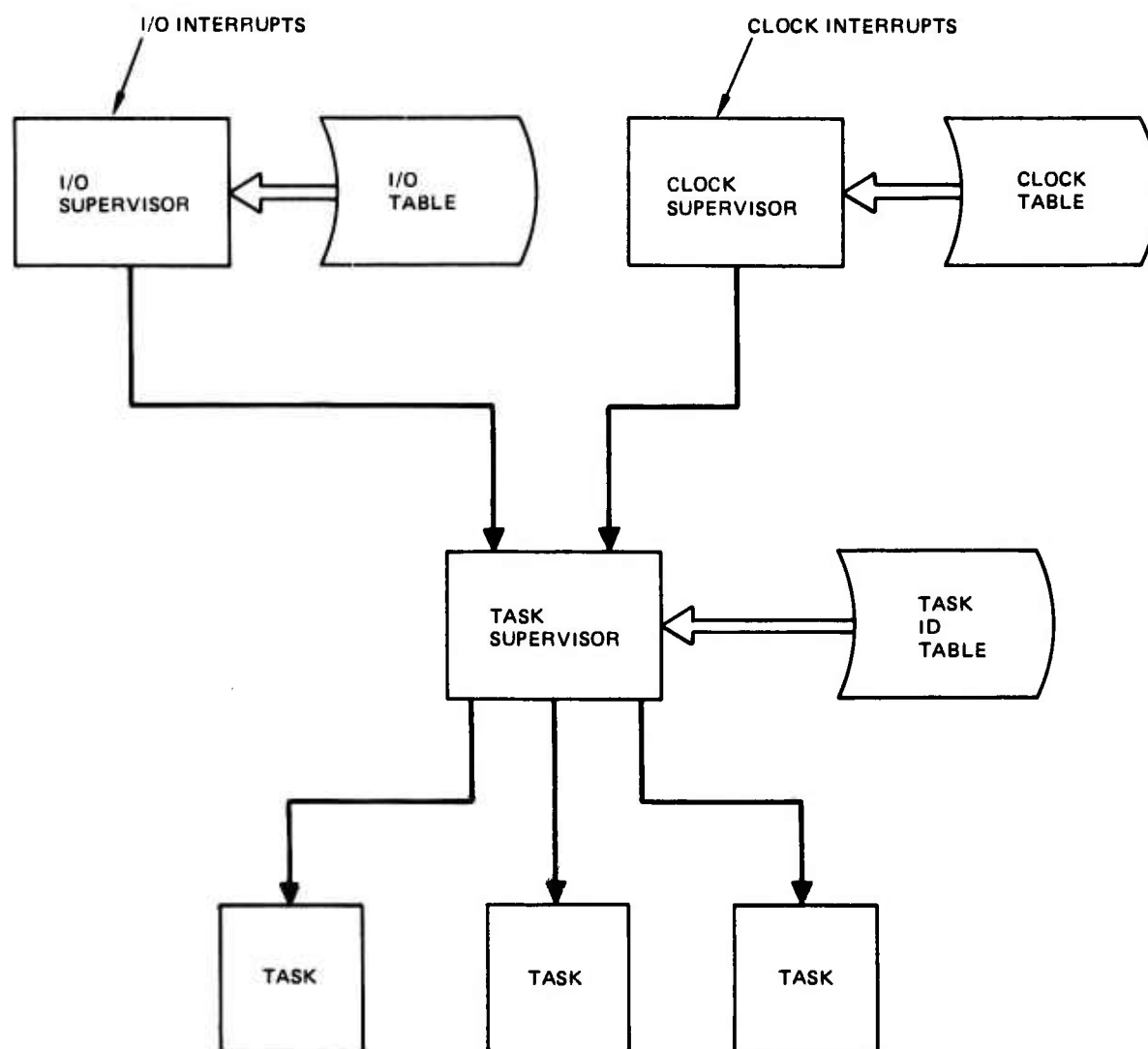


Figure 51. Executive Routines and Tables

define what task is to be executed for each interrupt received from an external device, for each block of data transmitted from an I/O device, and for each time strobe received from the system clock.

To illustrate the use of tasks and the executive, consider the following hypothetical situation. Some input device provides data that must be used in two different operations that are not related to each other except that they operate on the same data. The software would be partitioned into three tasks. One task, which will be called task A, would be responsible for getting the data from the input device, reformatting the data to a common form, and storing the data where it could be accessed by the other tasks. The other two tasks, which will be called B and C, would perform the required operations on the data. The sequence of events would be: The DP receives an interrupt from the input device indicating that data is ready. The interrupt routine in the executive would service the interrupt and discover that task A

was to be executed. The interrupt routine would tell the task supervisor part of the executive to execute task A. The task supervisor would look up task A in the task ID table to discover its location and priority. The priority determines which task will be executed first when several are waiting to be executed at the same time. When task A is the highest priority task waiting, the task supervisor will start it executing. Task A reformats the data from the input device and stores it, and then it tells the task supervisor to execute task B and C. Task A is now finished. The task supervisor will execute tasks B and C when they are the highest priority tasks waiting.

With this arrangement either task B or C could be modified or eliminated without affecting the other. And task A does not know where B and C are located, therefore, they can be moved about without changing A. If the input device is replaced by another device which formatted the data differently only task A would have to be changed to reformat the data.

When the tasks are placed in physical memory modules each module has an initialization section which begins at the first location in the memory module. This initialization section tells the executive what tasks are in this module, and what their starting locations and priorities are. During DP initialization the initialization section of each hardware memory module is executed. This lets the executive know where all tasks are located so that these locations do not have to be fixed for all configurations.

#### 7.4 DIGITAL PROCESSOR ARCHITECTURE

The architecture of the digital processor is concerned with the instruction set to be executed and the facilities required to support the instruction set. As discussed previously, these architectural considerations are strongly dependent on the functions to be performed, the processes involved in these functions, and the total weapon system characteristics. The specification of the processor instruction set and support facilities is required to determine the requirements placed on the hardware implementation of the processor by the system functions. The interaction of implementation cost with processor architecture was considered in the derivation of the instruction set and processor facilities presented in this section. These architectural parameters were used to derive the memory and throughput data presented in the previous sections of this report.

The processing requirements derived in this study are based on a general purpose instruction set. The diversity of processing functions and types of operations involved in these functions does not allow the development of a special purpose instruction set. The instruction set has been organized in five major categories as shown in Table 18. These categories are generic to all digital processors, but the support facilities for each instruction type in these categories have been defined in accordance with the system characteristics. The dynamic instruction

TABLE 18. INSTRUCTION CATEGORIES

CATEGORY	DYNAMIC EXECUTION, percent
DATA TRANSFER	37
INDEX REGISTER MANIPULATION	5
ARITHMETIC AND LOGICAL	20
TRANSFER OF CONTROL	37
SYSTEM CONTROL	1

execution mix in this system is similar to the mix encountered in other digital processors used for real-time process control. The mix shown in Table 18 was derived by analysis of the software pertinent to the CORE functions.

The data transfer instructions provide for all transfers of data among the arithmetic unit registers, index registers, and operand memory. Operands may also be furnished to the arithmetic or index registers by the program memory. All data storage elements required for interfacing the digital processor with the other weapon subsystems including the weapon bus are considered part of operand memory. Both direct and indexed addressing modes are required for all data transfer instructions involving operand memory, and all operand memory locations must be addressable in both modes. Sixteen index registers are required by the functions to be performed in the digital processor. Index registers are dedicated to the executive software functions to improve the efficiency of the executive software which is in series with all digital processor operations. A very large number of index registers would be required if registers were dedicated to the applications programs. The remaining index registers are shared by the applications programs, and require that their contents be saved and restored if one program is interrupted by a higher priority application program. To perform the save and restore function, data transfers between operand memory and the index registers are required. The data transfer instructions provide transfers of data between any two registers or between any register and any operand memory location.

The index register manipulation instructions involve the modification of the contents of the index registers. The primary uses of the index registers are address control for processing array data, control of iterative operations (loops), and argument transfer (indexed addressing) for subroutines. Approximately 70 percent of all arithmetic and data transfer instructions use the indexed addressing modes. Variable increment and decrement instructions are required for array data addressing. Occasionally, more complex index register data modification is required. Data transfers between the arithmetic registers and the index registers allow the full arithmetic instruction capability to be used for index register data modification. A decrement and branch on zero instruction is required for iterative operations.

A complete set of arithmetic and logical instructions is required for single precision operands. A single precision word length of 16 bits is adequate for the majority of the computations required by the DP processing functions. The higher precision computations are performed by double precision software routines, and require that computation aids be provided in the implementation and instructions using these aids be provided. Instructions which allow software floating point routines to be written are required to support the few computations with large dynamic range. Arithmetic instructions involving both arithmetic registers and operand memory are required. At least two general purpose arithmetic registers are required. Both indexed and direct addressing modes are required for instructions using operand memory. The multiply and divide instructions account for 3 percent of all instructions executed. This relatively low percentage is not an indication of the computational complexity of the functions performed, but rather due to the emphasis on weapon control by the processor.

The high percentage of transfer of control instructions in the dynamic instruction mix is a result of both the modular software structure and the decision processes involved in the real time control of the weapon subsystems. Many of the software modules are general purpose subroutines requiring subroutine call and return instructions. Both immediate and indirect subroutine address specifications are available. Indirect addressing allows many alternative subroutines to be called from a single program statement by setting up the branch address as a function of system state. An indirect branch capability is also allowed. A subroutine return stack with at least 32 levels is required to support software modularity and allow interrupt capability. Conditional branching instructions which test arithmetic computational status, stored function status, and the status of the other weapon subsystems are required for control of the weapon functions. The logic instructions in conjunction with conditional branching on arithmetic unit status provide system status assessment. The status of external subsystems is contained in operands either received via the weapon data bus or set as a result of weapon bus interrupts. However, status storage must be provided both to store computation function state from previous iterations of the function and for subsystems which directly interface with the digital processor, e.g., the weapon bus controller.

A vectored priority interrupt capability must be provided to synchronize the operations of the DP with the operation of the other weapon subsystems. Eight levels of interrupt are adequate not only for executive software, but also to provide for simple synchronization of operations concerned with other subsystems which may directly interface with the processor. The system control instruction category is concerned with software control of these interrupts and with the setting of the processor status storage states.

This instruction set defines the functional data and control transfer paths which must be present in the processor. However, the instruction set requirements must be considered in the context of the throughput requirements in the hardware implementation of the processor. The



dynamic instruction execution mix is useful in the optimum allocation of processor hardware resources to functional elements implied by the instruction set.

Many hardware structures may be considered for the hardware implementation of the processor, the optimum structure is highly dependent on the available technology. The two breadboard processors constructed on this program are implementation examples of the digital processor architecture discussed in this section. These breadboard processors do not implement the complete instruction set specified for the digital processor, but the differences are minor and are the result of evaluating the breadboard processor performance of the weapon system functions to determine the required instruction set. The throughput capability of the two breadboard processors for the specified instruction mix is: 1.85 MIPS (DP 1), 2.3 MIPS (DP 2). The throughput capability was derived by assuming equal frequency of execution for each instruction in a category to determine the average speed for each category, and then weighting these average speeds according to the specified instruction mix. The primary factor causing the difference in throughput capability is processor clock rate, but there are some instruction set differences also. Either of these breadboard processors meet the peak throughput requirements discussed in the previous section for the CORE functions.

## 7.5 DIGITAL PROCESSING SYSTEM REQUIREMENTS

In the preceding subsections, a digital processing system configuration was defined, the functions to be performed by the digital processor were identified, and digital processor architectural considerations were presented. Digital processor design parameters have been developed for a limited, but representative, set of weapon configurations.

The general purpose instruction set and support facilities identified for the digital processor are capable of performing all required computational and control operations for the weapon system. Memory size and throughput requirements have been developed using this instruction set to perform representative examples of the digital processor functions. The requirements derived so far are base requirements. That is, the throughput requirements were just that necessary to perform the defined CORE functions and the memory requirements pertain to a single configuration. It is now necessary to consider the total modular weapon system (including all configurations), and to determine an appropriate factor for growth.

### 7.5.1 Processor Throughput Requirements

The determining factor for throughput is the propagation delay requirement for the autopilot stabilization function. In subsection 7.2, the required processor operations to perform this function were given. Using the data given there one can determine the required processor

throughput. Let  $t_s$  be the time to perform a short instruction and  $t_L$  the time to perform a long instruction. Then let

$$t_L = K t_s \quad (1)$$

The throughput for short instructions is the inverse of  $t_s$ . The required  $t_s$  is given by

$$t_s = \frac{525 \times 10^{-6} - W T_w}{350 + N + 20 K} \quad (2)$$

where

$W$  = number of words transmitted on the bus during the time interval allotted to the process

$T_w$  = time required to transmit one word

$N$  = number of instructions required for system management and interrupt responses.

Now define:

$T_1$  = time spent doing system management functions

$T_2$  = time spent doing the stabilization computations

$T = T_1 + T_2 = 525 \times 10^{-6} - W T_w$

For the selected processing system configuration,  $N = 260$  and  $W = 9$ . In order to choose a value for  $T_w$ , examine Figure 47. Noting that required processor throughput capability is a monotonically increasing function of  $T_w$ , it is desired to choose  $T_w$  as small as is practical. It has earlier been noted that a bus transmission rate of 100,000 words per second is a practical goal for the modular weapon ( $T_w = 10$  microseconds). Higher rates can be achieved, but the figure does not indicate a substantial decrease in processor requirements by doubling the bus rate. Therefore,  $T_w = 10$  microseconds is chosen.

Using the above relations, the required throughput for short instructions has been plotted in Figure 52. This curve can also be interpreted as the mean throughput the processor must exhibit during the time interval,  $T_1$ .

Another quantity of interest is the mean throughput during the total interval,  $T$ . This is also shown in Figure 52. It may be noted that during this period the average instruction mix ratio is 3 percent longs to 97 percent short instructions.

To complete the picture, the mean throughput during the interval,  $T_2$ , has been calculated and is also shown in the figure. During this period, the instruction mix includes about 5 percent longs.

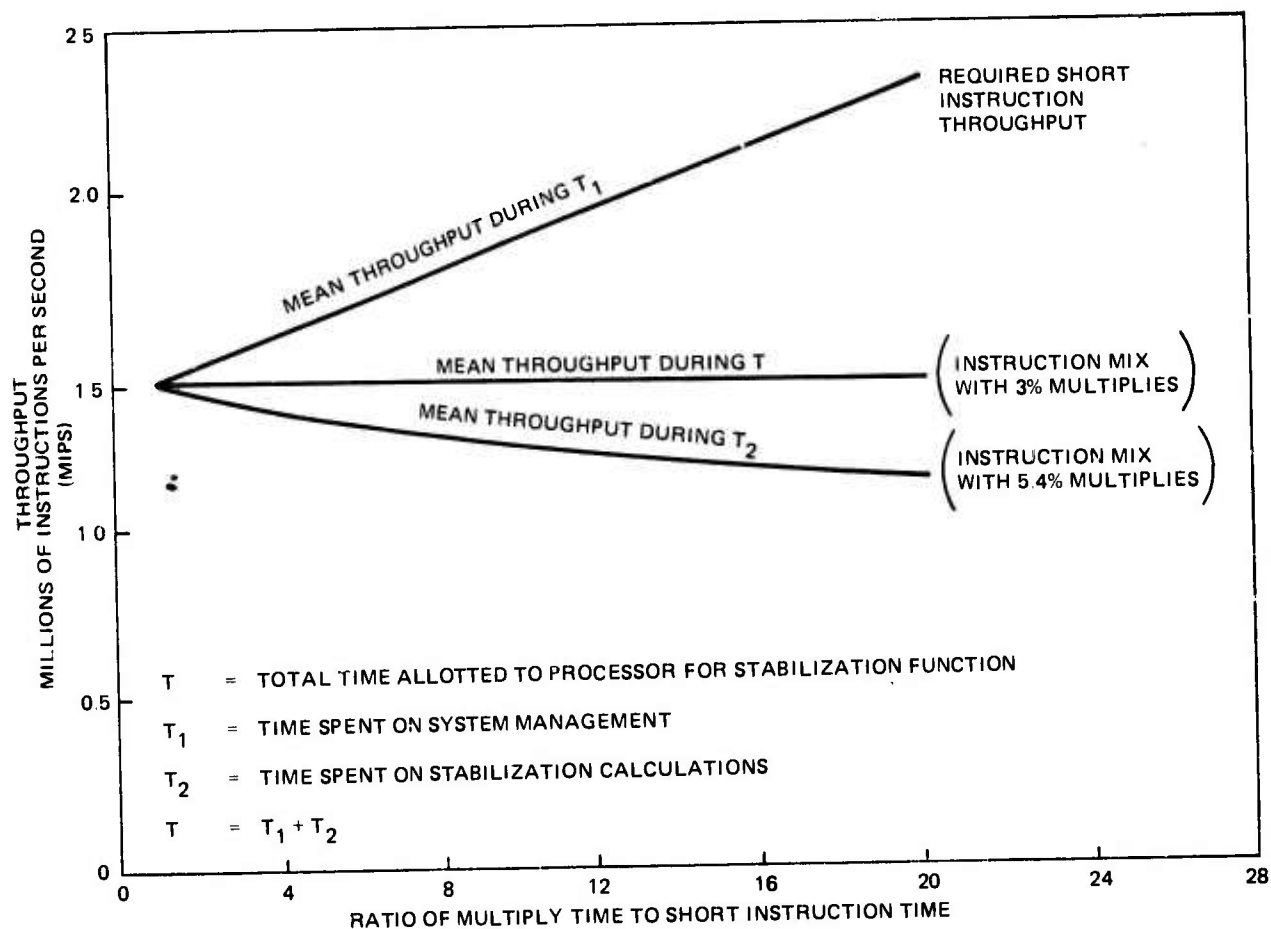


Figure 52. Peak Throughputs for Time-Critical Flight Control Stabilization Function

The expected average percentage of long instructions for the total CORE functions is about 3 percent as may be seen by referring to Table 17. Thus, the mix during the interval,  $T$ , is representative of the total CORE functions. Therefore, the  $1.45 \times 10^6$  instructions per second required on the total interval,  $T$ , is taken as the base requirement for the digital processor (with an instruction mix containing 3 percent long instructions).

There is always a bit of subjectivity involved in choosing a growth factor. Reference, again, to Table 17 shows a fairly comfortable growth factor with respect to average throughput. However, the peak throughput requirements over the weapon life may be greater than the identified value for several reasons. The principal stabilization function requirements affecting peak throughput are propagation delay and computational complexity. These parameters are primarily a function of the weapon aerodynamic configuration, and variations could produce a significant increase in peak requirements. In addition, other weapon configurations may have a second time critical function. Within the CORE function definition, the most likely function of this type is a

critical data transfer which may be required during the stabilization loop computation period. If an interrupt is received from an external subsystem during this period, the peak throughput requirement is increased by approximately 16 percent, even if the implied task is not immediately executed. This type of interrupt does not affect aerodynamic stability if its frequency of occurrence is low, but this cannot be guaranteed for all weapon configurations. In any case it appears risky to commit to a design in which there is no apparent margin. It is felt that a margin of at least 50 percent over the base requirement is a good practice. (This would not be an adequate margin for a requirement based on average throughput, however.) The base requirement is 1.45 MIPS. Therefore, with a 50 percent margin above the base, the recommended requirement is 2.2 MIPS with the given instruction mix (3 percent long instructions).

The above throughput requirement should be checked for compatibility with average requirements. Table 17 gives the average as just under 900 KOPS. This is a 15 percent margin above the base which should be adequate. It should be noted that the above requirements are contingent on achieving a weapon bus transmission rate of 100 K words per second.

#### 7.5.2 Processor Program Memory Requirements

The program size requirements developed previously only pertain to a single representative weapon configuration. A growth factor must be applied to account for computational complexity variations over all weapon configurations. The adequacy of this type of program memory space specification depends on weapon assembly procedures, since it implies that only the software pertinent to the particular weapon configuration is contained in the digital processor. The digital processor program memory must be non-volatile, since its contents must be retained from the time of software loading until completion of the weapon mission with no external power applied to the weapon. Furthermore, the memory implementation must be compatible with the processor throughput specification.

The most obvious program memory implementation which allows each weapon to only contain the pertinent software is a read/write memory which is loaded during weapon assembly. However, the only read/write memory technology of sufficient speed is volatile and would, therefore, require power to be applied continuously after weapon assembly. Hierarchical memory systems using a combination of slow and fast read/write storage elements can provide a non-volatile program memory, but imply high complexity and cost.

A read-only program implementation provides the required speed and is non-volatile. However, to achieve minimum program space would require that unique program memory elements be available for each weapon configuration. The appropriate memory elements could either be inserted in the processor at the time of weapon assembly or be permanently installed in a processor which could then only be used

with a single configuration. Neither of these options are desirable because of the implied proliferation of configuration items and corresponding weapon configuration control problems.

Of the options discussed above, a non-volatile memory loaded at weapon assembly time does not violate the modularity concept since it is assumed that the weapon does undergo a systems test at the end of weapon assembly and the appropriate program could be loaded at that time. The option with the read-only memory modules for the appropriate mission loaded at weapon assembly does violate the modularity concept.

Since the hierarchical memory does have some system attractiveness, it is worthwhile examining the cost question a little further. Military, bipolar, random access memories of the requisite speed are expected to cost about eight times read-only memory costs, bit for bit. (See Section IX.) Thus, even neglecting the cost for the non-volatile storage, the hierarchical system would have to reduce total memory requirements by at least a factor of eight to become attractive from an economical point of view. This is not likely.

Since none of the above memory options can be wholeheartedly recommended, it is necessary to base memory address space requirements on some other approach that is both practical from a system standpoint and economically attractive. Such an approach is to store the entire program for all configurations in read-only memory. As indicated in Section IX, the cost trends extrapolate to about one tenth of a cent per bit in the early 1980's. The packaging density is high so that required board space will be small. This approach does make system program changes somewhat more difficult, however, considering the state of technology, it is the most practical approach. Memory address space requirements are based on it.

The identified storage requirement for the CORE functions is 3000 words (see Table 17) for a single configuration. Since the weapon system is not well-defined at the present time, it is difficult to make an estimate of program requirements for all configurations. However, an estimate can be made of the additional memory required for adding a new configuration to the existing system. It was estimated in Section IV that adding a new airframe would increase the program requirement for the flight control by less than 300 words. Adding a new mid-course guidance mode can increase the strapdown inertial subsystem requirements by requiring a new program to filter the update data. The filter requirement in the base program is about 600 words. The new filter could require substantially less if some of the routines of the first filter can be used. It could require more if additional states were required. As an estimate, let us use 600 words.

It is estimated that adding a new subsystem will, on the average, increase the system management program by 30 to 40 words. The recommended system design has 16 terminals on the weapon bus. Two of these are dedicated (DP and avionics). Therefore, a configuration

change could introduce 14 new subsystems. This would add about 600 words to the system management. Adding the above gives a total of 1500 words additional program memory for adding a completely new configuration. The precision of the above number is open to question, but it is probably more precise than an estimate of the number of configurations that will be in the system.

An alternative way to estimate program size is by estimating the probable number of additional major subsystems that would be added to the base configuration. Table 19 shows an attempt at this. The two estimates presented above are not intended to convey the impression that an accurate assessment of program memory size has been made. However, two conclusions can be drawn:

- Additional configurations can easily add 4000 words to the base requirement
- By the time 4000 words are added one has a very complex modular system.

In other words, a total of 8000 words of program memory is enough for a rather complex modular weapon system. Clearly, additional subsystems can be added without limit, but it is doubtful if operational considerations can justify a weapon system with more alternatives than are given in Table 18 or which has more than four completely different configurations. Applying a 100 percent margin to the above estimate to account for uncertainties and for growth gives a minimum of 16 K words for program address space. It is stressed that this is a minimum requirement. While this amount of memory would not be installed in initial systems, the processor must have provision for adding memory as the system grows.

TABLE 19. ESTIMATED INCREASE IN PROGRAM MEMORY REQUIREMENTS

GENERAL SUBSYSTEM	NUMBER ADDED	ADDITIONAL PROGRAM REQUIREMENTS (words)
TERMINAL GUIDANCE	6	240
MIDCOURSE GUIDANCE	4	2400 + 160
AIRFRAMES	3	900
MISCELLANEOUS ADDITIONAL SUBSYSTEMS	10	400
TOTAL		4,100



### 7.5.3 Processor Operand Memory Requirements

The operand memory must include provisions for all computational variables and constants pertinent to the software contained in the processor. All variables are stored in read/write memory elements, and blocks of memory can be dedicated for each class of functions performed by the processor, thus providing sufficient capacity for all weapon configurations within a minimum memory space. As previously discussed, read/write memory technology with speed compatible with the throughput requirement is volatile and, therefore, subject to having its contents modified by power transients. The majority of the variables can be accidentally changed with minimal effect on weapon performance. However, some parameters are critical, e.g., target location for the inertial reference navigation function, and must be protected. A non-volatile read/write memory of 256 words provides sufficient capacity for mission critical parameters and allows system recovery in the event of power transients. The operand memory space which must be provided for storage of constants depends on the method of loading the software in the processor. However, just as in the case of the program memory, read-only memory would appear to be the most practical approach with present technology. Memory address space requirements are estimated on this basis. Reference to Table 17 shows that the required operand memory, for a single configuration, is about one half of the program memory size. The largest part of this is constant memory associated with the various subsystems. It may be expected that constant storage requirements will grow at the same rate as program memory as new subsystems are added. Requirements on read/write memory will grow at a slower rate, but it represents a smaller part of the total requirement than the constants. Based on the above considerations, the ratio of required operand memory to required program memory should decrease somewhat. However, it is not expected to decrease enough to allow a 4 K address space to have sufficient margin. Therefore, an operand memory address space of 8 K words, minimum, is specified.

### 7.5.4 Weapon Bus Requirements

The characteristics of the preferred weapon bus were presented in subsection 7.1. Here, some performance related parameters are specified.

#### Bus Word Rates

A weapon bus word rate of 100 K words per second is required to be compatible with the specified processor throughput and system communication requirements for time critical functions. The data bus rate is based on 16 bits data content for data words. This data content is compatible with both the processor word size and the data word size of the Stores Management System interface with the avionics. Thus, format conversion complexity is minimized at these prime interface points. Most other weapon subsystem parameters have less than 16 bits quantization, allowing all message words to have a common format. Subsystem data can either be transferred in packed format or by transferring an appropriate number of fill bits in the 16 bit data



field, depending on desired subsystem data format. The word rate capability is required for each of the communication paths (data and interrupt). The average word rate on each bus for representative weapon configurations implies a low bus occupancy factor and, therefore, good response for critical transfers.

#### Transmission Reliability

The transmission reliability requirement is based on allowing no more than one word error per 100 missions, on the average. For a typical weapon mission of 1000 seconds duration and an average bus rate of 10 K words per second, it is required that the probability of word error be less than  $10^{-9}$ . A word error is defined as accepting, as correct, a word with one or more erroneous bits.

Considering that the majority of the bus transfers are associated with data which is periodically renewed, the occurrence rate of missile critical errors is much lower than one per hundred missions.

### 7.6 CONFIGURATION DEPENDENT PROCESSING FUNCTIONS

Besides the CORE functions, there are a number of other weapon functions which affect, and are affected by, the CORE processing system. These functions differ from one weapon configuration to another and are called configuration dependent functions.

Some of these functions could be performed in the digital processor. However, whether or not they are performed in the processor, there are data transmission requirements associated with them. While these requirements have no measurable effect on required processor throughput capability, they do affect memory requirements. These functions are discussed below in the context of the three weapon configurations.

Candidate configuration dependent processing functions from the point designs of Section V for the three weapon configurations are identified. The requirements placed on the digital processor (DP) by each function are shown for two conditions: performing the function in the DP, and performing the function in another weapon subsystem. The effect of the added requirements on the baseline design requirements and the effect on the other weapon subsystems are discussed for each configuration.

#### 7.6.1 Weapon Configuration I

In addition to the CORE functions, the inertial-body coordinate conversion and two different portions of the correlation function of the RAC subsystem were identified in the point design. The DP requirements associated with these functions are shown in Table 20.

The inertial-body coordinate conversion computations can easily be performed by the baseline DP configuration with no increase in requirements. If this function is not performed in the DP, a general purpose processor is required in the RAC subsystem. Although the

TABLE 20. DP REQUIREMENTS FOR RAC SUBSYSTEM FUNCTIONS

FUNCTION	LOCATION	PROGRAM SIZE	OPERAND MEMORY	THROUGHPUT, KOPS		BUS RATE (words/sec)	
				SHORT	LONG	DATA	INTERRUPT
I-B COORDINATE CONVERSION	RAC	10	-0-	5.8	-0-	1400	100
	DP	260	85	91	8	3200	114
CORRELATION INTERFACE ②	DP	110	30	685	98	35K	1090
CORRELATION INTERFACE ③	DP	70	15	300	-0-	11.6K	360
CORRELATION	RAC	10	0	0	-0-	4	1

performance requirements on that processor are minimal, RAC subsystem cost would be increased.

The correlation function computations could be performed by the DP for either interface definition (slight throughput increase for interface ②) if total average throughput were the only consideration. However, the iteration rate for both interface definitions is commensurate or higher than the iteration of the critical stabilization function. The correlation computations for interface ② would require additional operations during the critical propagation delay period and force a substantial increase in DP throughput capability. Selection of correlation interface ③ would result in a smaller increase in DP throughput capability, but is, nevertheless, undesirable. Therefore, it is recommended that the entire correlation processing be performed within the RAC subsystem.

#### 7.6.2 Weapon Configuration II

The configuration dependent functions identified in the point design are the LORAN position processing, the data link message decoding, and the line and field signal processing for the EO seeker. The DP requirements associated with these functions are shown in Table 21.

Both the LORAN positioning processing and EO data link message decoding present a minimal addition to the CORE function requirements. The position processing would require a general purpose processor within the LORAN subsystem which cannot be justified. By performing the message decoding function in the DP, the data link interface with the weapon bus is greatly simplified and the difference in DP requirements is negligible. The combination of line and field processing for the EO seeker cannot be performed in the DP without a large increase in DP throughput. This is primarily due to the high iteration rate of the line processing. The field processing computations can easily be performed by the DP with the baseline configuration definition. However, the field processing interface for which these requirements have

TABLE 21. DP REQUIREMENTS FOR LORAN, DATA LINK, EO SEEKER FUNCTIONS

FUNCTION	LOCATION	PROGRAM SIZE	OPFRAND MEMORY	THROUGHPUT, KOPS		BUS RATE, (words/sec)	
				SHORT	LONG	DATA	INTERRUPT
POSITION PROCESSING	LORAN	10	-0-	0.2	-0-	5	1
	DP	120	50	3.2	0.01	5	30
MESSAGE DECODING	DATA LINK	10	5	8.0	-0-	270	30
	DP	110	20	25.0	-0-	270	30
LINE + FIELD	DP	250	40	1150.0	37.0	83,500	2625
FIELD ONLY	DP	215	40	28.0	1.1	1380	60
LINE + FIELD	EO	20	-0-	12.0	-0-	240	60

been derived is not compatible with existing EO seeker implementations. Only the requirements shown for both line and field processing in the EO seeker are germane to configurations using existing EO seekers.

### 7.6.3 Weapon Configuration III

The configuration dependent functions identified in the point design are the data link message decoding, and the line and field processing for the IIR seeker. The DP requirements associated with these functions are shown in Table 22.

The requirements for the message decoding function are identical to those presented for Configuration II. The average throughput requirements for performing both the line and field processing in the DP are compatible with the baseline DP capability. However, the iteration rate of the line processing results in an increase in the number of operations required during the critical stabilization function propagation delay period. Consequently, the DP throughput capability must be increased to accommodate this peak load. The DP requirements

TABLE 22. DP REQUIREMENTS FOR DATA LINK, IIR SEEKER FUNCTIONS

FUNCTION	LOCATION	PROGRAM SIZE	OPFRAND MEMORY	THROUGHPUT, KOPS		BUS RATE, (words/sec)	
				SHORT	LONG	DATA	INTERRUPT
MESSAGE DECODING	DP	10	5	8	-0-	270	30
	DATA LINK	110	20	25	-0-	270	30
LINE + FIELD	DP	250	40	315	10.7	26,200	800
FIELD ONLY	DP	215	40	28	1.1	1,380	60
LINE + FIELD	IIR	20	-0-	12	-0-	240	60

associated with IIR field processing and the discussion concerning this function for Configuration II are also applicable to this weapon configuration.

#### 7.6.4 Generic Classes of Configuration Dependent Functions

The examples of configuration dependent processing functions discussed in the preceding paragraphs provide insight concerning the relationship between the characteristics of the functions and their effect on digital processor requirements. The characteristics of primary interest are the function processing bandwidth, computational complexity, and its interfaces with all other weapon functions.

The functions of the weapon subsystem can be arbitrarily divided into two classes on the basis of processing bandwidth, which determines the required iteration rate of the function of it as performed by the digital processor. For the purpose of this study, all functions with an iteration rate commensurate with or higher than the stabilization function iteration rate (400 Hz) will be classified as signal processing, and the lower iteration rate functions will be designated as data processing. Computational complexity is concerned with both the number of input parameters and operations on these parameters which are implied by the function. The interface characteristic of primary interest is the location of the destination of the function outputs relative to the function inputs.

Most signal processing functions are characterized by relatively simple operations on large amounts of data. The small program size and high throughput requirement exemplified by the E-O seeker line processing is typical for performing this class of function in the digital processor. As a result, the throughput capability of the processor may be exceeded due to a combination of functional computational complexity and software operations to support data transfers. Although the major functional outputs of this type of processing function are steering signals for flight control, a number of parameters usually must be sent back to the data source subsystem to control its operations, but the principal effect on weapon bus capacity is the input data rate.

Data processing functions usually involve relatively complex operations. The throughput requirement to perform this type of function in the digital processor is generally small compared to the total for the CORE functions due to the low function iteration rate. The program memory size requirement for this class depends not only on the computation complexity but the similarity of the operations involved in the function to the operations of the CORE functions. A relatively complex function may have minimal program memory requirements if existing subroutines can be used for the function. Most candidate data processing functions have a major interface with the CORE functions.

#### 7.6.5 Weapon System Considerations

Each configuration dependent processing function must pass certain criteria if it is to be performed by the digital processor. At

the lowest level, only the cost of implementing the function in the subsystem versus the digital processor need be considered. The subsystem cost tradeoff must include not only the implementation cost of the function itself, but also relative costs of conforming to the standard interface definition. The subsystem cost differential must at least balance the cost of DP memory associated with the function, assuming sufficient processor throughput to perform the function. Cost comparisons of this type are commonly used for functional partitioning in point designs, but the modular weapon system characteristics require additional criteria.

Even in point designs, the combination of all cost effective functions (determined on an individual basis) may exceed the digital processor capability requiring additional tradeoffs to determine the relative cost of increased processor performance versus dropping some desired functions. There are obvious problems in extending this procedure to the modular weapon system. If a function is performed by the digital processor in any weapon configuration, then the processor must be capable of performing the function in all pertinent weapon configurations in addition to the CORE functions. The processor requirements were established in the previous subsection by applying a growth factor in the CORE function peak processing requirements for the selected weapon configurations. This growth factor provides for configuration dependent variations in the CORE function peak requirements over all weapon configurations. This implies that a configuration dependent function which increases the peak CORE function requirements may not be allowed for all weapon configurations. Thus, the decision process for each configuration dependent function must consider the effect of that function on CORE processing requirements for all pertinent weapon configurations in addition to the requirements for the function itself. This decision process, therefore, involves the evaluation of total processing requirements for all pertinent weapon configurations and requires that the function fit within the excess processor capability after the CORE functions are performed in each configuration. In general, only functions in the data processing class will meet this criterion.

The desirability of performing any configuration dependent function should be a function of its differential implementation cost weighted by its probable percentage usage over all weapon configurations used. The frequency of occurrence of the function is especially important if the processor contains software pertinent to all weapon configurations. A function with large program memory requirements which is only pertinent to a small percentage of weapons would generally require a large implementation cost advantage per use to overcome added memory costs over all configurations.

No specific rules or guidelines have been developed which may be generally applied to configuration dependent functions. However, the important factors which must be determined in the decision process have been discussed.

## SECTION VIII

### TECHNOLOGY STUDY

This task was concerned with using some of the newer technologies to implement the DP processing system. The goal was to achieve the DP performance requirement with a significant reduction in cost, power dissipation, and size relative to available digital processors today. In particular, three technologies were to be investigated:

- Low power Schottky TTL (LPSTTL)
- Silicon-on-sapphire C-MOS (SOS C-MOS)
- Integrated injection logic ( $I^2L$ )

The study focused on the use of LSI devices, in one or more of the above technologies, to implement the several DP functions. There are several approaches to the use of LSI in the DP system:

- Use of commercially available LSI designs
- Semicustom LSI design (e.g., gate arrays or cell arrays)
- Custom LSI design.

The SOS C-MOS and  $I^2L$  processes are still too new to have any significant number of commercial LSI designs available or even viable semicustom approaches. Hence, custom LSI approaches were considered for these technologies. In the LPSTTL process, on the other hand, there are available both commercial LSI design and semicustom LSI design approaches. Therefore, the LPSTTL investigation was confined to these two approaches.

LPSTTL is the most mature of three technologies and would certainly be a strong contender for implementing the DP system if it were to be done today. There are available, now, a fairly good selection of LPSTTL LSI devices designed explicitly for computer or digital processor applications. Among the more interesting of these devices are the so-called bit-slice micro-processor chips (also called microcontrollers and various other names). These are essentially a slice through the major functions of a computer CPU and may be 2 bits wide or 4 bits wide at the present time. Thus a single 4-bit slice alone would be a 4-bit microprocessor. Four of them would form a 16-bit microprocessor. Typically, other devices need to be added to complete the CPU function or to give better performance. Among the added chips are Microprogram Control Units (MCU), ROMs for the microprogram, carry look ahead generators for faster arithmetic, and various registers and multiplexers. Using a typical four-bit microprocessor slice, a 16-bit microcomputer CPU can be built with approximately 20 chips. While such a processor cannot satisfy the DP requirements, it is far more powerful than any available MOS microprocess-based computer.



The use of the bit-sliced microprocessors is not limited to the kind of application described above. They can be used in more powerful designs by adding additional peripheral devices and/or paralleling functions. By these techniques a processor satisfying the DP requirements can be built. This was the goal of the DP-X design study reported herein.

## 8.1 DP-X DESIGN STUDY

### 8.1.1 Design Approach

The purpose of the DP-X design study was to design a digital processor that satisfies the DP requirements and uses available LSI, LPSTTL, bit-slice microprocessors. The DP-X design will be compared to other approaches in the cost analysis study to determine how successful this approach is.

The general performance requirements are given below along with some rather general ground rules.

#### DP-X PERFORMANCE REQUIREMENTS

- 16 bit fixed point
- $\geq 8K$  program memory address
- $\geq 4K$  data memory
- $\geq 2$  arithmetic registers
- $\geq 16$  index registers
- $\geq 32$  level pushdown stack
- $\geq 2.8$  MIPS (short equivalent)
- Block transfers
- -180 instructions
- Satisfy DP interrupt procedures

#### DP-X Design Study Ground Rules

- Modular design
- Minimize power consumption
- Minimize size

This task was undertaken to show that with low power Schottky (LPS) circuits, a near term design can be implemented that establishes the validity and practicability of the proposed specification. To achieve performance comparable to higher powered circuits with the low powered circuits, increased parallel processing (i. e., pipelining) is required. Larger scale integration is required to reduce component package count. Modularity is necessary to prevent premature obsolescence and to allow continued update of hardware modules without changes in software.



The DP-X design is carried only to a point sufficient to count cycle times of various instruction types and understand programming implications. This baseline design also provides mechanical packaging information so that realistic manufacturing cost and physical size can be determined.

#### 8.1.2 Instruction Formats

A key factor in any processor design is the instruction format. With increasing recognition of software development and maintenance costs, an easily understood and usable format is very important. With so-called standardization more frequently heard, a popular format might have some side benefits. Another assumption is that with ROM density increasing, small differences in program memory size may not be a significant factor in the 1980's.

A number of formats were examined including the ones in DP I and II. The basic decision to be made was whether to use a fixed instruction word length or a variable word length. By stringent adherence to the minimum requirements the operations code requires 8 bits; arithmetic register, 1 bit; index register, 4 bits; memory address, 13 bits; with a total of 26 bits. This also allows no expansion in memory size beyond 8000 words, except by paging. A variable instruction format such as IBM or Interdata shown in Figure 53, which consists of either 16 or 32 bits, is very popular. The break even point in efficiency between fixed and variable format occurs where 62 percent of the instructions are 32 bits; beyond that the variable format requires more program bits. The variable format, however, allows almost unlimited memory expansion as far as missile requirements is concerned, and also has the happy coincidence of each 16 bits corresponding to the data word length of 16 bits, a convenience in system layout. Therefore, the variable word length was adopted. An attempt was made to use the exact Interdata instruction set so as to take advantage of existing software support. This appeared to be impractical for missile applications because too many compromises must be made. The final set adopted, as shown in Figure 54, actually includes the Interdata format. A third register field is added to the address portion optionally so that by not using that field, it reverts to the Interdata format. Greater flexibility and versatility can be achieved in instruction formatting. The 4-bit register fields in these instruction formats happen to fit the 4-bit LSI arithmetic logic units, ALUs, now becoming available from more than one vendor. With appropriate multiplexing circuits and microprogramming control, implementation of Interdata instruction set is also possible. The new 4-bit ALUs have 16-word internal registers. The match with the 4-bit register field is perfect. This allows the use of one set of ALUs with 16 general purpose registers or two sets of ALUs with 16 arithmetic and 16 index registers separately. For speed reasons, two sets of ALUs are required.

There are two popular and almost comparable 4-bit slices on the market, the AMD 2901 and the MM 6701. They differ only in the instruction format. The AMD 2901 allows a three address operation

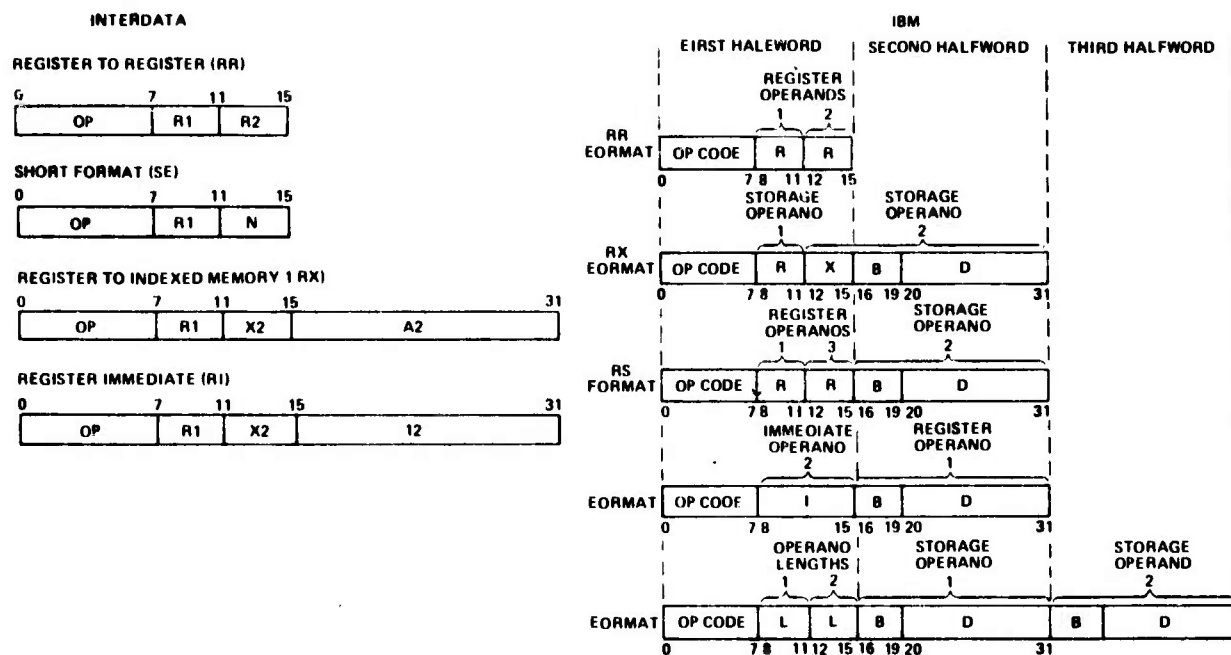
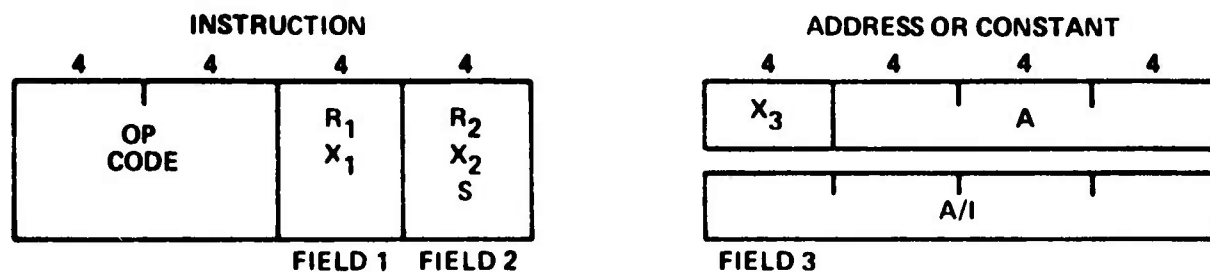


Figure 53. IBM and Interdata Formats



- R ARITH REGISTERS 0-15
- X INDEX REGISTERS 0-15
- A ADDRESS 16 BITS (OR 12 BITS)
- S CONSTANT 4 BITS
- I CONSTANT 16 BITS

INSTRUCTION MAY OR MAY NOT BE FOLLOWED BY ADDRESS

ALL COMBINATIONS IN FIELDS PERMISSIBLE

EXAMPLE:  $(A + (X_3)) + (R_1) \rightarrow R_1; (X_3) + S \rightarrow X_3$

Figure 54. DP-X Instruction Format Includes Interdata or IBM Format

the way the macro-instruction set is configured whereas the MM 6701 does not. For this reason, the AMD 2901 was chosen as a nominal part for this design exercise.

### 8.1.3 Typical Instructions

The 3-register address format is illustrated to demonstrate its versatility (Figure 55). First are the short instructions of one word (16-bits) which may be register-to-register (R for arithmetic and X for index registers), or register and a short constant (S). For the long instructions, the next word is an immediate constant (I), an address (A) or a register field (X) and an address (A). In the last case, only 12 bits of address are available but the index register has 16 bits, so the total range is still 16 bits. Another useful type of instructions is the block transfers. Here the S field may be the number of words,  $X_1$  may be the beginning register location, and  $X_3, A$  are the indexed memory addresses. This type is highly desirable where frequent interrupts occur and registers must be preserved for later resumption of interrupted routines.

In view of the separate ALUs used, two different sets of arithmetic instructions are required. In this construction, the index arithmetic is designed to be integer and the real arithmetic fractional.

### 8.1.4 DP-X Units and Buses

DP-X (Figure 56) is organized in modular fashion so that data follows the pipeline structure to achieve the speed required. There are two almost identical arithmetic units (can be made completely identical), one for indexing and the other for regular arithmetic, with their respective register fields. Each arithmetic unit has complete microprogram control so that it is only necessary to pass from one to the other the undecoded operation code and the requisite register fields. Interrupt is shown as a separate unit, but is actually packaged together with the memory controls and other miscellaneous control circuits. The I/O as shown is intended to be that portion of the bus interface unit. It is assumed to tie in the data memory bus, but can be tied to any other bus if more convenient.

### 8.1.5 DP-X Pipeline

The pipeline flow (Figure 57) starts with the program address control where either the next program word is set up or a branch address is used. This address fetches the word in program memory. The first one is always an operation code, which goes to the operations register (OPR). The next word goes to the address or constant register (IXAR), and may be an address or constant, or may be another operation code. In the latter instance, the content in IXAR is simply ignored, and on the next cycle the same information is read into the OPR. The code in OPR is decoded through the microprogram control memory, and the control information then is available in the pipeline register simultaneously with the address availability in IXAR.

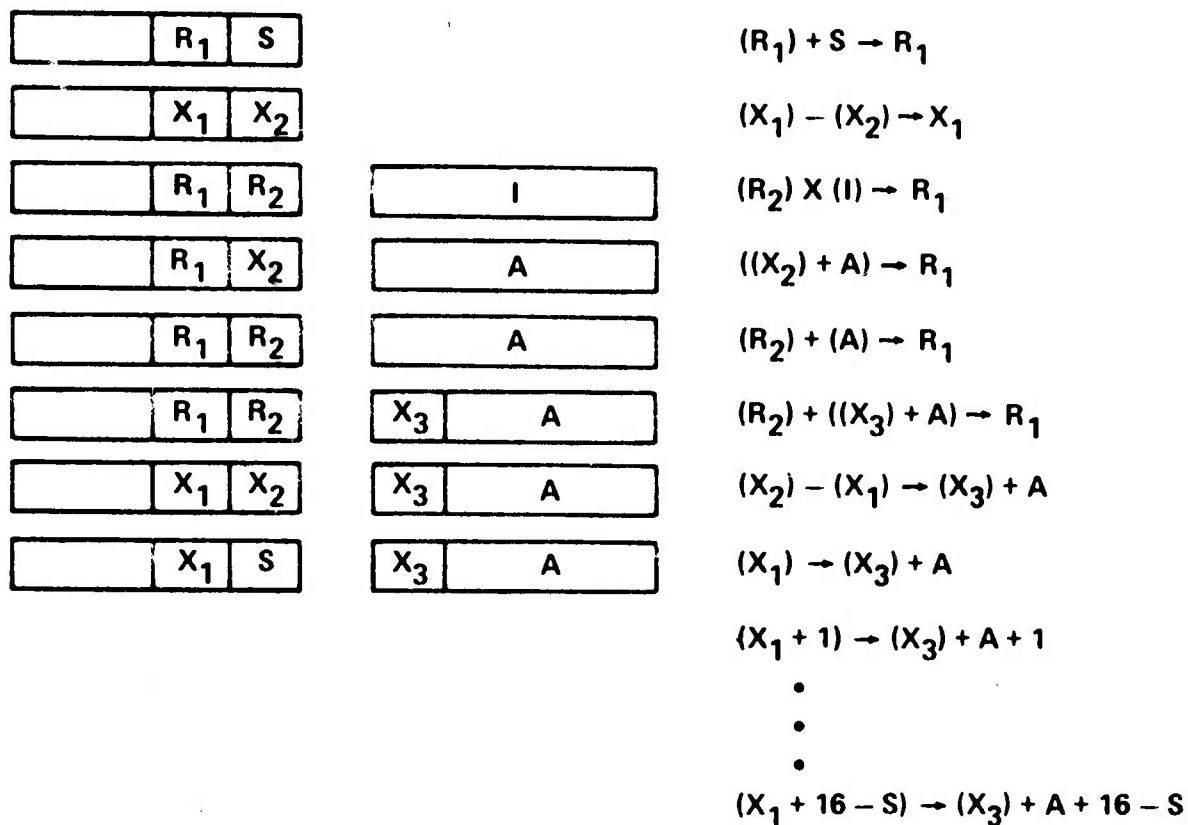


Figure 55. Typical Instructions

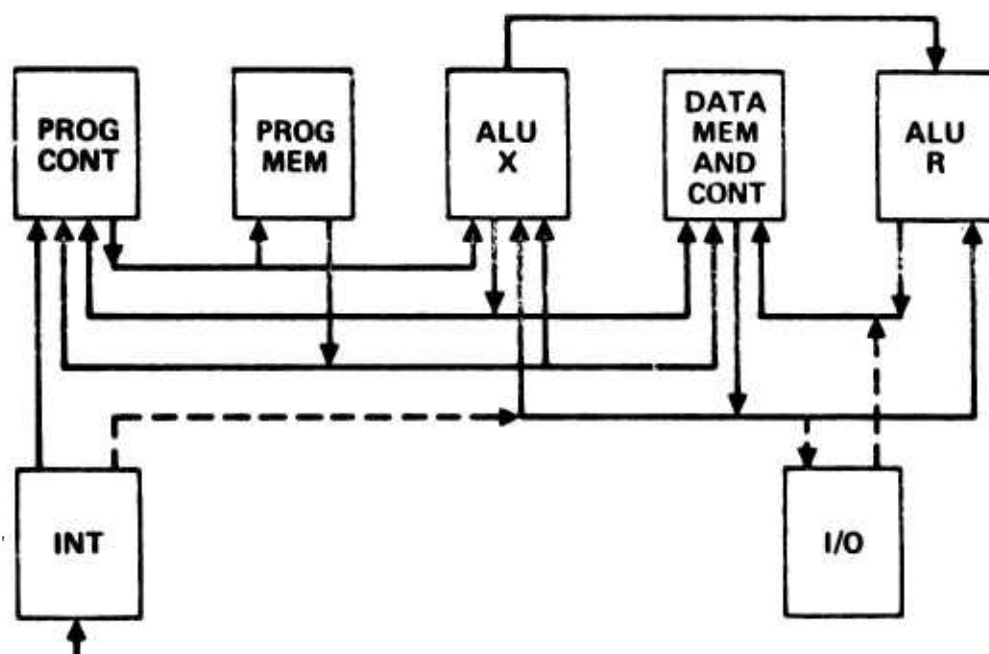


Figure 56. DP-X Units and Buses

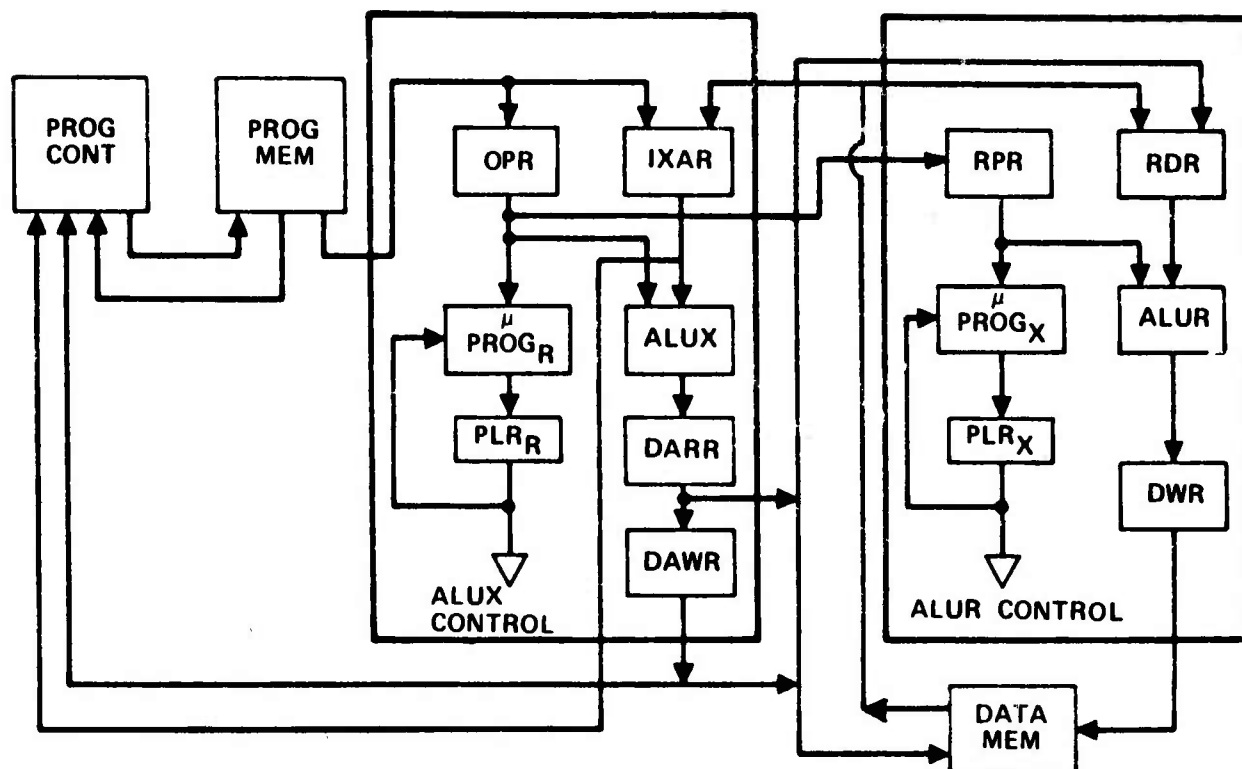


Figure 57. DP-X Pipeline

The arithmetic chips in ALU X then take two cycles to complete a short arithmetic or simply passing it through. In most cases, this is the address for the data memory, which takes one cycle to fetch the data. Simultaneously, the operations code and register addresses are passed along to ALU R so that when the data becomes available, the microprogram control information for ALU R is also available. In another two cycles, ALU R finishes its operation and if the output is to be stored in memory, it is deposited in the data write register (DWR). Previously, the address from ALU X was in data read address register (DARR); it is passed along to data write address register (DAWR) so that the information for writing into the data memory is again available at the same time.

For most short arithmetic instructions, two cycles are all that is necessary. For multiplications, handshake logic stops the pipeline until the ALU can again handle the next instruction. For branches, the pipeline is stopped if and when an irrevocable operation is about to be performed, which may be in the wrong branch. For that reason, if branch is to be effected, it sometimes takes one or more cycles to be completed in addition to the regular two cycles. This can be shortened by one cycle if a small amount of hardware is added to recognize certain branches.

The length of the pipeline is further illustrated by the timing chart (Figure 58). A span of 8 cycles is possible. It should be noted that

PAC	0	0A	1	1A	2	2A	3	3A
OPR		0		1		2		3
IXAR		0A			1A		2A	
ALUX		0			1		2	
READ DM					0		1	
RPR			0		1		2	
RDR						0		1
ALUR						0		1
WRITE DM								0

Figure 58. Normal Cycle

each cycle is basically a memory access time plus certain register and multiplexer delays. For bipolar memories and low power Schottky register circuits, it is estimated that for worst case military temperatures of 125°C, a cycle time of 150 to 175 nanoseconds should be allowed if the faster bipolar ROM or RAM are used.

The arithmetic chips have a cycle time slightly longer than the projected memory access times. Certain status bit transmission time through the control circuits must also be allowed for branch or other control decisions. The arrangement of using two memory cycles for one arithmetic cycle is a sufficient build-in safety factor. On the other hand, it is possible to increase the multiplication or division speed by adding a faster clock cycle by adding more circuits. If really high speed multiplication is necessary, a serial-parallel multiplexer appears to be practicable, but is not considered in the present baseline design.

#### 8.1.6 Program Control

Figure 59 illustrates DP-X program control. The basic element in program control is the program address counter (PAC). Parallel load of this counter via a multiplexer effects the various branch modes, including interrupt. For return to subroutine only, a 256 address stack is provided to avoid timing conflicts with the data memory. No facility for end of stack is provided because of the apparent infinite depth.

Not directly related to program control are the handshake logic for the two ALUs. The basic principle is to determine the state of readiness to transmit and receive of succeeding units in the pipeline organization.

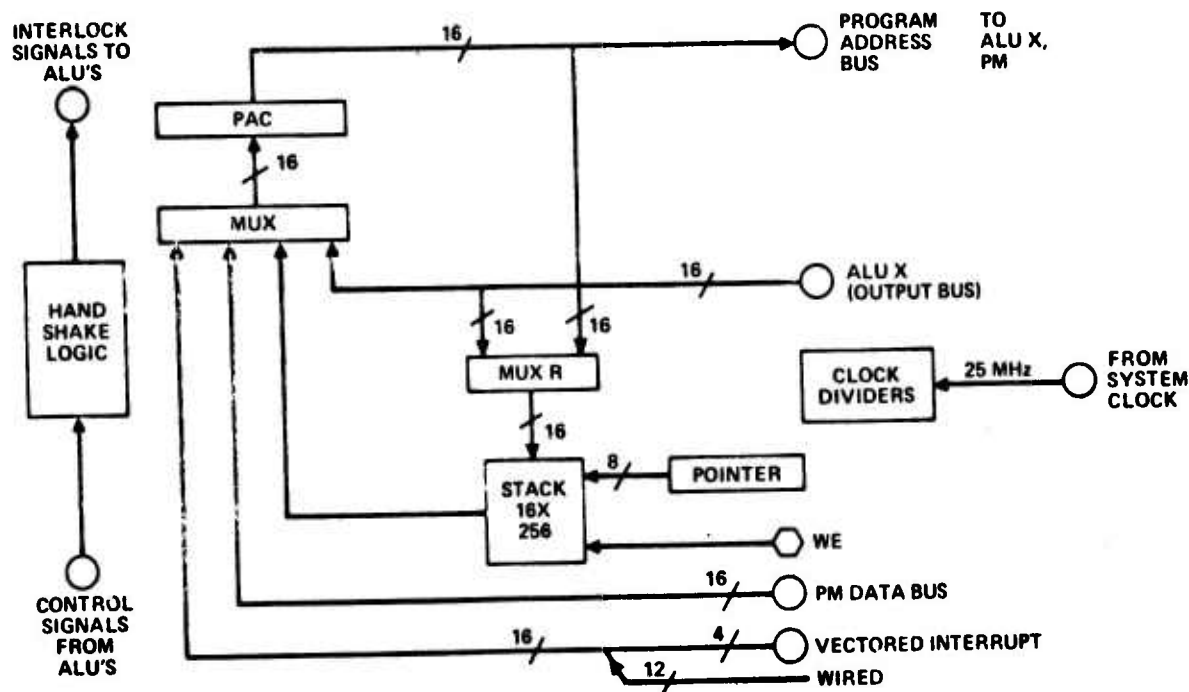


Figure 59. Program Control

The basic clock of nominally 25 MHz is divided down one to four to provide a system clock pulse of 25 percent duty cycle. This is further divided down a factor of two to provide the A and B clock pulses of the two ALUs, whose cycles are staggered one system clock time. High powered Schottky circuits are used in the clock distribution system to effect minimum clock skew between different circuits.

The address processor (ALUX), shown in Figure 60, is representative of the two ALUs and is the more complicated one. The program data bus splits to OPR and IXAR. The OPR is part of a simple micro-program control built with ordinary multiplexers. Available sequence controls were found to be not too suitable because features such as stacks are not too useful in a high speed machine where steps are minimized. Necessary features such as branch address input multiplexing are not always available. Some high powered (logically) sequence controller chips have awkward addressing arrangements that waste control memory space and some multiplication and division control features are not directly usable. A more appropriate sequence controller is described later.

The 16-word registers in the ALU chips dictate the system design to a considerable extent. In view of the still limited number of registers, each time an interrupt occurs a number of registers must be preserved for later resumption. This is anticipated with the block transfer instructions.



### 8.1.7 Comparison of DP-1 and DP-X Speeds

A few selected passages of DP-1 program were checked on DP-X by trial programming. Results are shown in Figure 61. The message parameter update module is branch intensive. The task queue dispatch is block transfer intensive and the filter routines are computation (especially multiplication) intensive. The time ratios reflect the advantages and limitations of DP-X. The time ratios were computed using the pessimistic limit of 175 nanoseconds. If the more optimistic limit of 150 nanoseconds is used, the ratios become, respectively, 1.08, 1.43, 1.1, and 0.93.

### 8.1.8 DP-X Packaging

Since the address processor is the most complicated unit, it was selected as an example in packaging layout. Dual-in-line on multiple layer printed circuit board was tried, but it quickly became evident that both cost and size are unacceptable. Hybrid packaging yielded

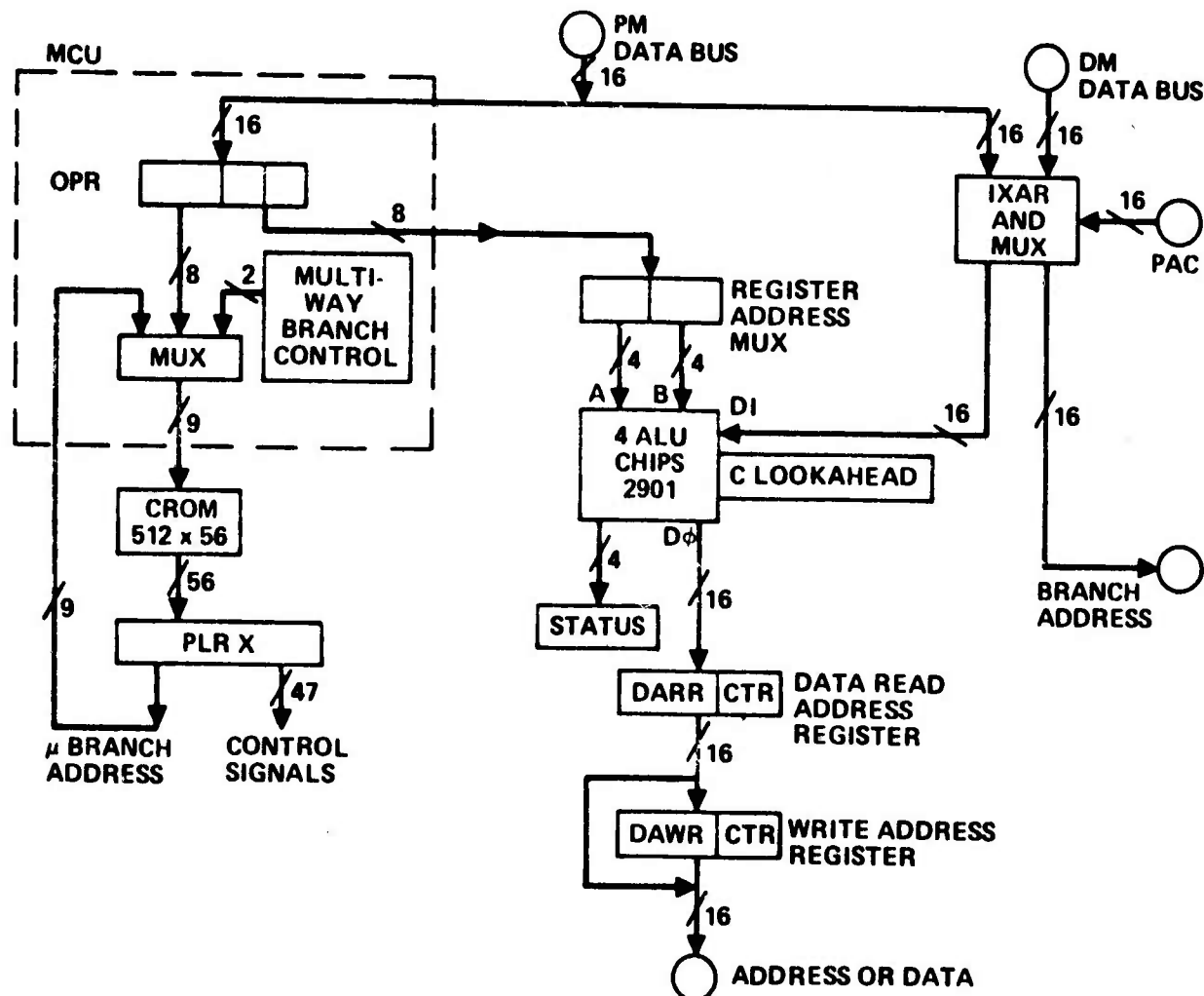


Figure 60. Address Processor (ALU-X)

ALGORITHM	MICROSECONDS		TIME RATIO DP I/DP X
	DP I	DP X (175 NS CP)	
MESSAGE PARAMETER UPDATE (MOREMO) MESSAGE STACK LENGTH (K) = 5	17.2 = $10 + 2.4 (K-2)$	18.4 = $8.9 + 2.1 (K-1) + 0.35 (K-2)$	0.93
TASK QUE DISPATCH (DISPAT) TASK QUE LENGTH (N) = 3	57.0 = $9 + 16 N$	46.5 = $9.3 + 12.4 N$	1.23
2ND ORDER FILTER	33.9	35.6	0.95
1ST ORDER FILTER	19.6	24.6	0.80

Figure 61. Comparison of DP-I and DP-X Speeds

satisfactory package size, but the integration of so many high level chips in a partially tested fashion introduces rework problems not completely predictable at this time. The last scheme tried appeared to be the most satisfactory. This is the packaging of each chip in square leadless chip carriers, of which 3M is the principal manufacturer. These leadless chip carriers are used in large quantities for commercial products. They are much smaller than comparable dual-in-line packages, and are designed for reflow solder onto ceramic circuit boards. A multiple layer ceramic board is envisioned and is also available from the same manufacturers. The density is adequate, comparable by hybrid packaging, and the ability to individually burn-in and thoroughly test each component prior to assembly is an important advantage. The cost of the leadless frame is comparable to equivalent hermetic dual-in-line packages, and may be even slightly lower. For this reason, switching by industry to such a package or equivalent is probably inevitable when high level integration with their many pinouts becomes commonplace.

Figure 62 illustrates the DP-X address processor in its proposed packaging.

The method of attaching a heat sink to the multilayer ceramic board has gone through several iterations. The adopted concept, shown in Figure 63, is to attach the board to the heat sink and mount the connector to the heat sink, similar to the SEMS approach. The heat sink then provides the mechanical connection (and heat path) to the interconnection plane. The heat sink also includes screw jacking arrangements for securing the board and for its removal in a controlled way. The force on the large connector is considerable and is not carried by the ceramic board.

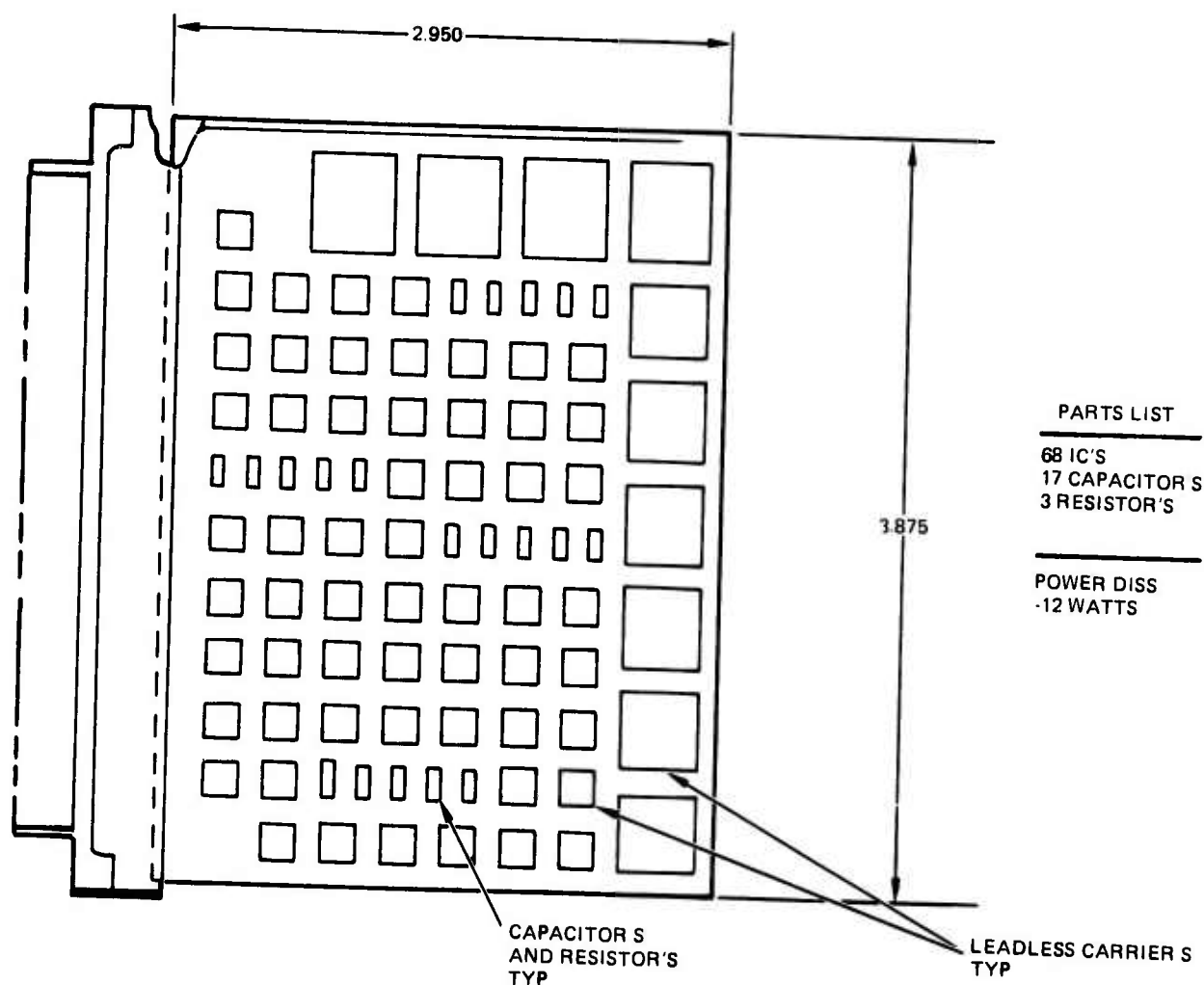


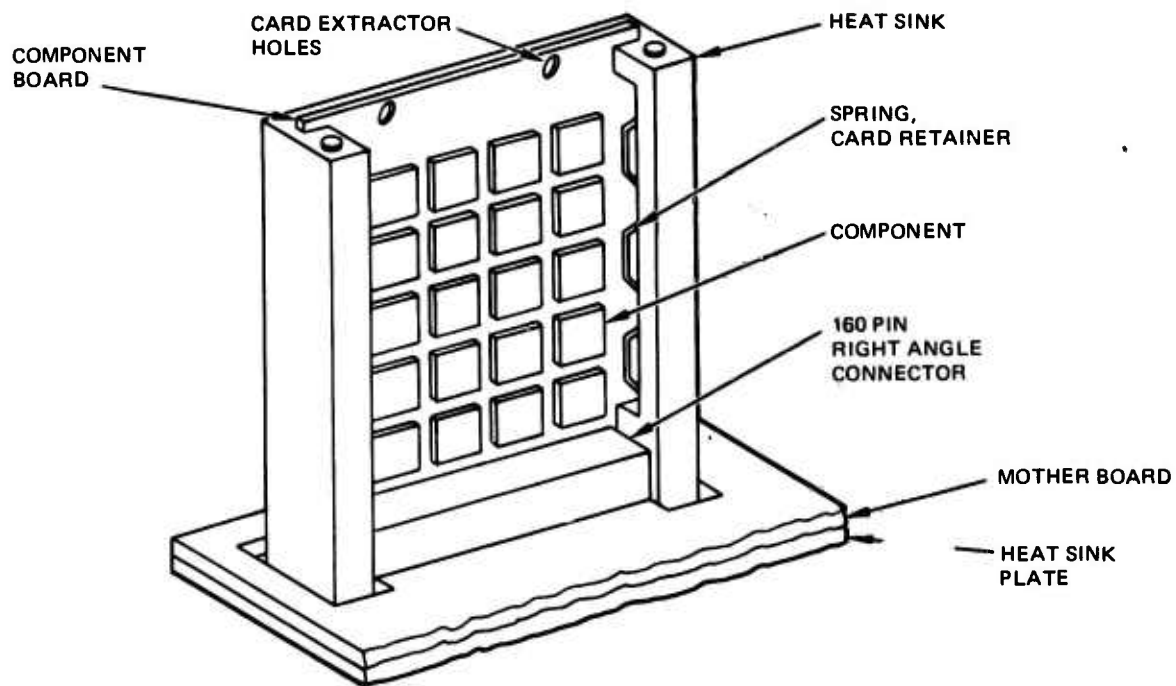
Figure 62. Advanced DP Address Processor

Flexible heat paths to ultimate missile structures can be made on top or bottom or sides by additional mounting plates after all connections are mated. This design concept is basically suitable for simple heat sink fabrication by extrusion, and does not indicate a costly part.

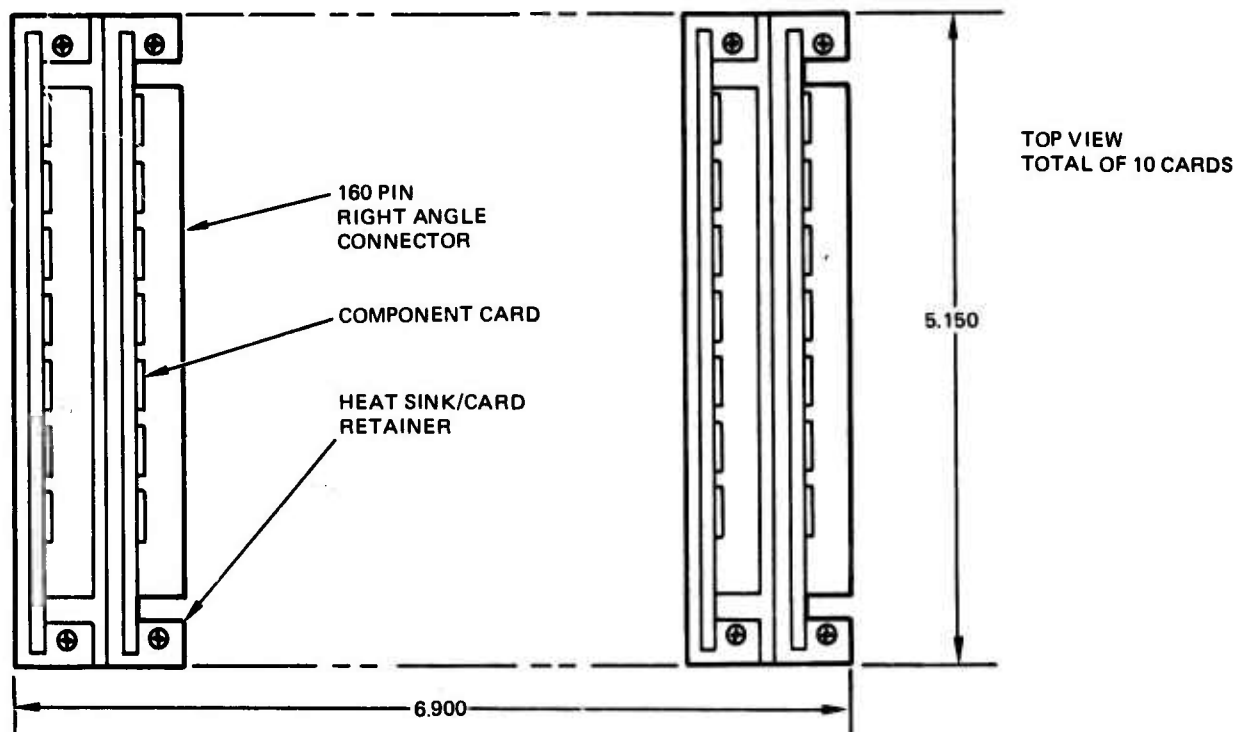
A concept of total size is shown in the 10-card arrangement. [Figure 63b]. There are three cards for the processor unit; three for memories, two for the portion of bus interface units peculiar to the processor, and two as spare slots. The overall size appears to be sufficiently small for most missiles in the modular weapons family.

#### 8.1.9 Summary

The DP-X design study indicates that an acceptable DP-X can be implemented with LPSTTL components now on the market. Two factors contribute to this particular design: (1) the pipeline organization remedies the slow circuit speed; (2) the leadless carrier packaging reduces the total volume. The component count is satisfactory although



a) CUTAWAY VIEW



b) TOP VIEW

Figure 63. Advanced DP

not comparable to full LSI custom designs. It should be noted that the cell array approach used elsewhere in estimating BIU circuits has not been applied to the CPU. A factor of 4 to 6 component count reduction may be expected in CPU if cell arrays are used.

A summary of DP-X characteristics follows:

- Low power Schottky (mostly)
- 2 ALUs: ALU X integer arith and address
- ALU R fraction arith
- ~2.5 MIPS short instructions
- ~30 W ALUs and memory control
- 16 registers each ALU
- Pipeline organization
- Standard instruction format
- Leadless carrier and multilayer ceramic boards
- 193 ICs on 3 boards (164 can be reduced with cell arrays)

## 8.2 LOW POWER SCHOTTKY CELL ARRAY TECHNOLOGY

There are several semicustom design approaches available in low power Schottky. Gate arrays have been available for several years. In this approach, a standard chip with several hundreds of unconnected gates is processed. An LSI circuit is created by interconnecting the gates with one or more layers of metalization. A somewhat different approach to semicustom LSI was examined in this study. It is called cell array semicustom LSI. In this approach, the basic building blocks are standard MSI functions rather than gates. The designer has available a library of standard functions (such as registers, multiplexers or a set of buffers). Up to ten of these cells can be placed on a single chip to create an LSI circuit. (The upper limit of number of cells per chip is determined primarily by yield considerations.) The designer has freedom with respect to selection of cell types and relative position of cells. Interconnection is by up to three layers of metalization. The cell array approach has a rapid turnaround time: about 12 weeks for design and processing. Cost studies show that acquisition costs are about the same as (or a little more than) buying the equivalent discrete MSI chips. However, overall costs tend to be lower because of decreased assembly labor. Moreover, a significant saving in board area is accomplished.

The cell array approach was applied to the design of the bus interface unit to examine its potentialities. The basic cell library used was that created by Hughes, Newport Beach. One new cell design was needed: an array of differential line receivers and drivers. The existing circuitry for the three cards (slave, interrupt and bus race) was partitioned into nine cell array chips. The only circuits not placed in

the cell arrays were the proms. With this approach, the three BIU cards can easily be packaged on a 3x4-inch ceramic circuit board. (It was assumed that the chips were placed in 64 pin leadless carriers.)

This technology could also be applied to the DP-X CPU. Of the 193 ICs used in that design, 164 could be replaced with cell arrays to achieve a reduction in chip number by a factor of 4 to 6. This would probably eliminate one circuit board.

The LSI cell arrays were also used in the I/O design and other circuitry in the six configurations of the cost analysis.

A photograph of a 4-cell array is shown in Figure 64. The results of the study indicate that this technology can be useful in the DP application for I/O and other peripheral circuits.

### 8.3 SOS C-MOS LSI TECHNOLOGY

Silicon-on-sapphire (SOS) technology significantly improves the performance of MOS circuits. The isolation achieved by the sapphire reduces stray capacitance which allows improvements in both speed and density. It is now known that a high performance, 16 bit CPU on a single chip can be achieved. An example is the Hughes SOS C-MOS processor which is illustrated in Figure 65. This processor had a design goal of 0.5 microsecond for add time and 3 microseconds for multiply. Preliminary tests on the processed chips indicate that the performance will not be too far off from the goal. It should be noted that achieving this kind of performance is not a straightforward matter with the SOS C-MOS process. The circuit and logic design required to achieve the high speed are considerably different from those normally encountered using discrete parts. For example, in the present design, it was desired to obtain a fast multiply to match the requirements for many missile applications. Because of the relatively slow MOS circuits, this was not possible without resorting to a carry-save multiply circuit. The conclusion is that a rather high performance processor can be achieved with SOS C-MOS, but only by exercising a great deal of care in the initial logic design and the circuit design and layout. There must be a considerable amount of interaction between these phases, i. e., the logic design must look ahead to the circuit problems that will be encountered in trying to get the desired high performance.

It is believed that the performance goal of the Hughes SOS C-MOS processor is close to what can be obtained with the process. In fact, the goal should probably be relaxed somewhat to give more design margin. It would appear, then, that the SOS C-MOS technology can probably not produce a single chip processor that will satisfy the DP requirement. However, it does appear that the requirement can be met with a multi-chip design. One approach would be a multiprocessor using two or more CPU chips similar to the one described above. This approach was used for one of the configurations in the cost analysis.



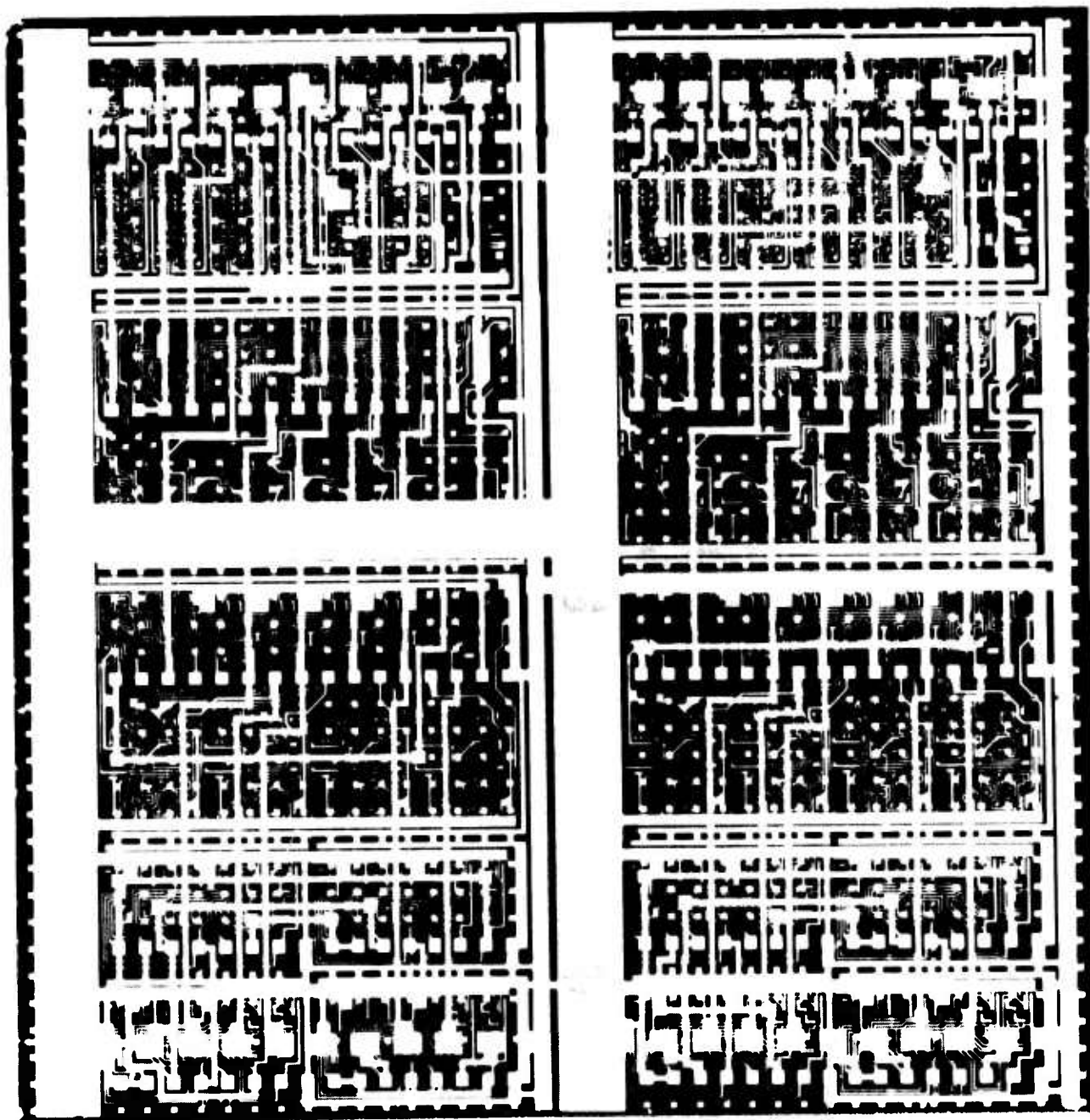


Figure 64. LSI Cell Array



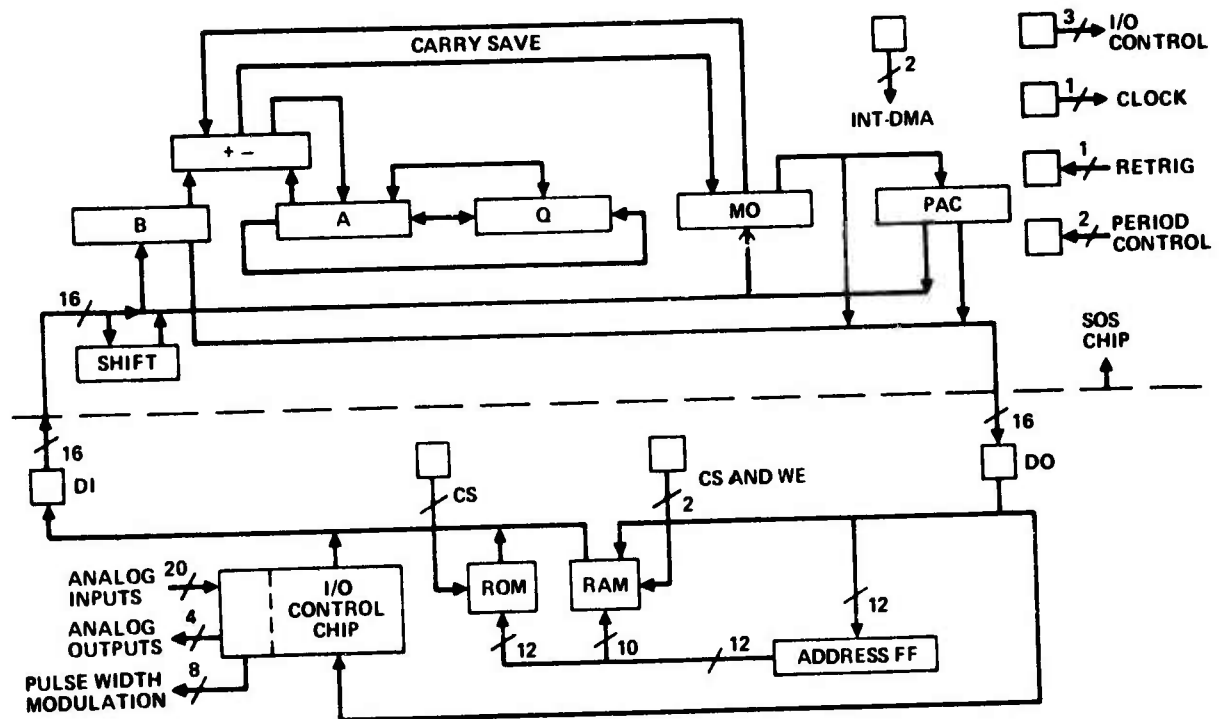


Figure 65. SOS-CMOS (SOCM) Processor Internal and External Elements

#### 8.4 $I^2L$ TECHNOLOGY

Perhaps the most promising of the new technologies is integrated injection logic ( $I^2L$ ). The potentialities of this process (not yet achieved in production) include a power dissipation in the MOS range but speed in the bipolar range. Moreover, the circuit design process is simpler than with SOS C-MOS since the effects of stray capacitance are not so devastating.

The technology is still rapidly evolving, and there would seem to be as many different  $I^2L$  processes as there are researchers working on it. Improvements added to the basic process include one or more Schottky diodes added to the gate input or output to increase speed.  $I^2L$  is the most likely of today's technology to obtain a DP-type processor on a single chip.

To explore the possibilities of the process for DP, a test design was made. It was noted before that none of the microprocessor control units (MCU) on the market meet the high speed requirements of DGWT processor design. Most available chips emphasize the looping or subroutine stacking requirements in micro-program and most frequently ignore the branching input multiplexing requirements. In a highly pipelined, high speed computer design, the objective is to reduce the number of microprogram steps to a minimum (one) in each ALU.

Another frequently ignored feature for high speed arithmetic are the shift controls and decision inputs required for multiplication and division. All the above functions are available in one chip or another, but not in the same chip. As a result, the selection of any available LSI MCU chips does not materially reduce the chip count, but frequently suffers in speed because of the need to serially stack multiplexing circuits external to the control chip, in addition to those already existing internally.

A design was defined (Figure 66) that is more suited to the short microprogram sequences encountered in DGWT type high speed computers. The branch address and instruction entering address ports are equally treated so that the two parts can act independently in a pipeline organization. In addition, a subroutine return register allows a one level subroutine, which is quite adequate in the DP design. An additional register for the next address allows larger flexibility in using the branch address. Decoders and control bit registers complete the design.

A cell library approach was taken for the chip design. A standard D register was modified so that a slice of the four-way multiplexer is included to allow easier interconnections. (It should be noted that the two levels of metal interconnection available greatly facilitate the chip layout.) This basic cell is used more than 40 times in the MCU chip for a 10-bit chip organization.

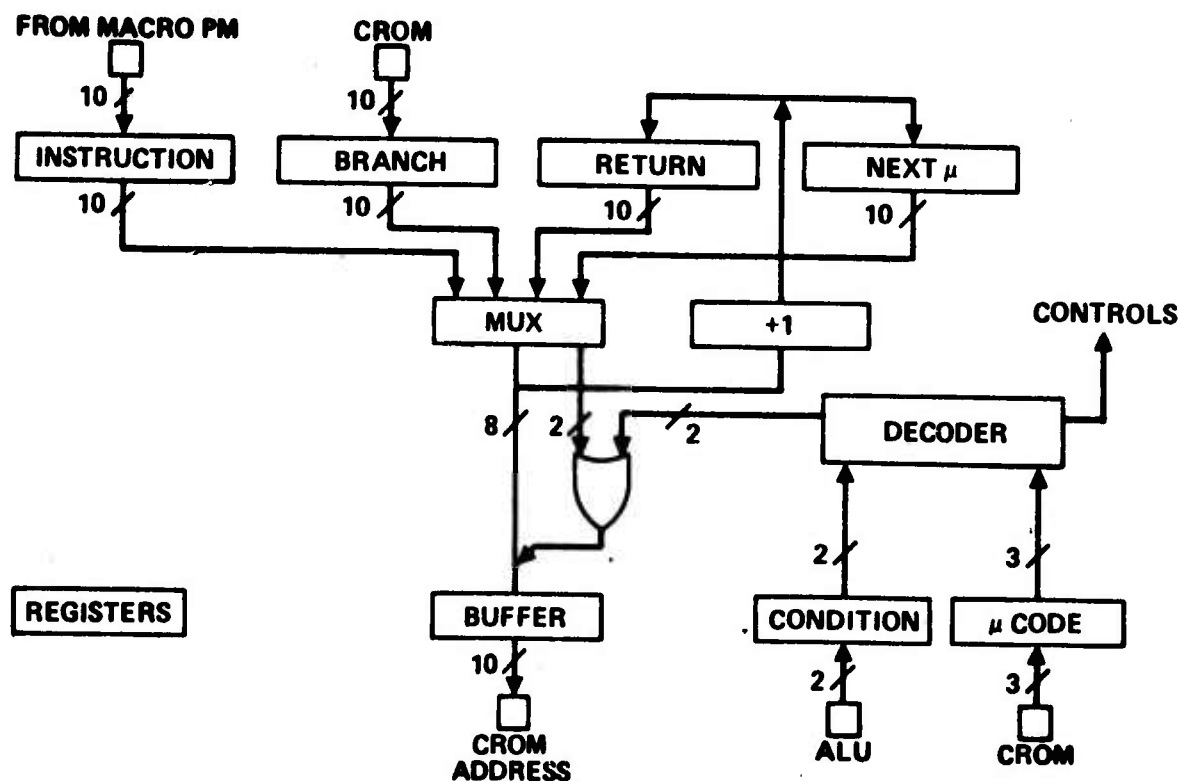


Figure 66. MCU Chip

The D flip-flop cells are stacked one above the other so that the injection rails run horizontally and bit lines of each register run vertically. There is no interconnection problem and the majority of the connections can be placed over the flip-flop cells.

The complete chip size, as shown in Figure 67 is considerably larger than the active logic area because of the number of bonding pads required. It can be concluded that considerably more circuitry could be placed on the chip. This design study illustrates that a high density can be achieved with this process even with a cell library approach. Moreover, the layout design rules followed in the design were for a process having a gate delay significantly less than 10 nanoseconds. It is clear that the I<sup>2</sup>L process would be an excellent choice for a custom design DP processor.

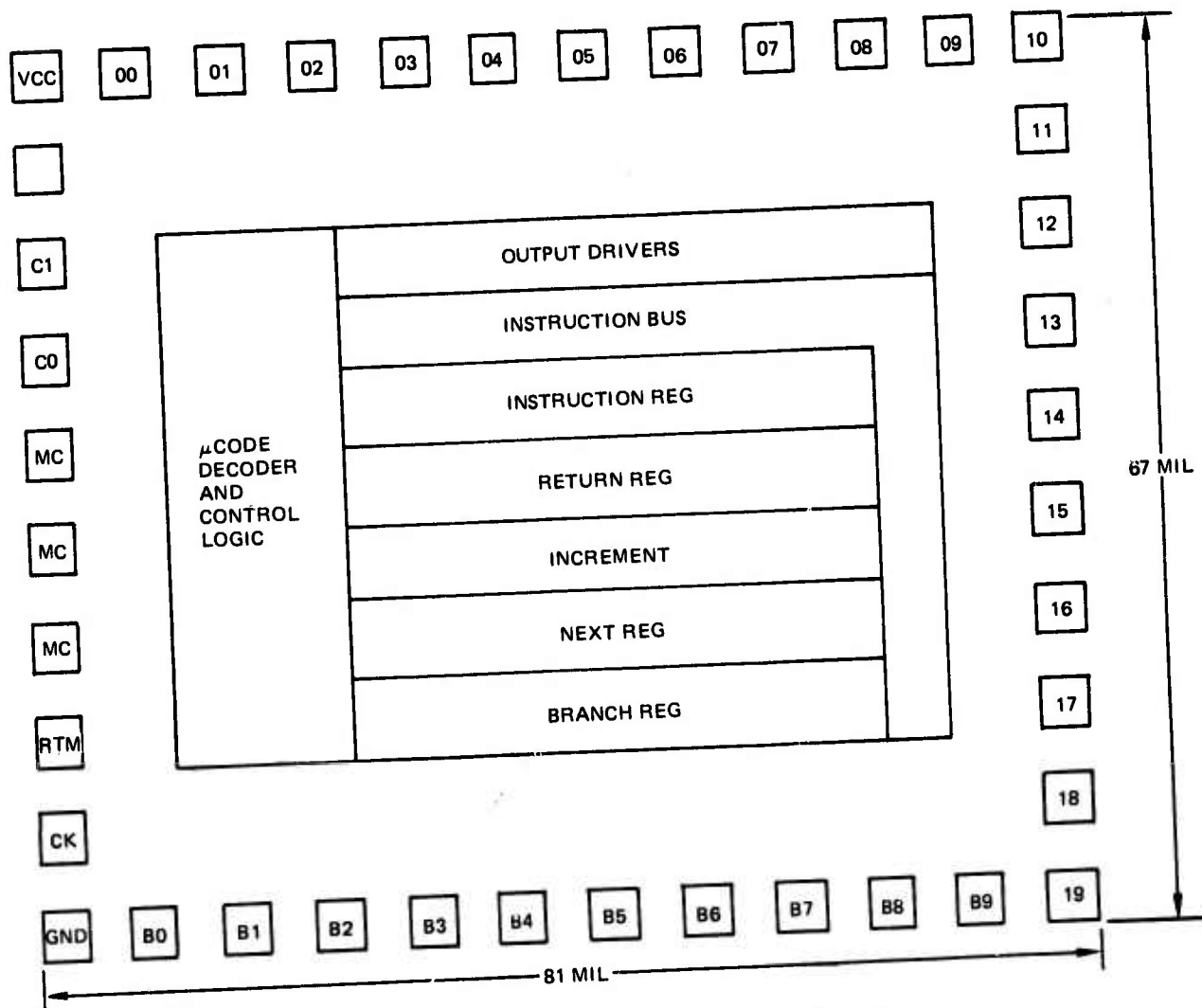


Figure 67. Microprogram Control Chip

## 8.5 TECHNOLOGY CONCLUSIONS

The fact that most microprogram control unit chips on the market are not very suitable for high speed computer organizations points out that as level of integration increases, less commonality between different users can be found. A highly integrated computer chip for missiles may not necessarily find general use in commercial fields, and the reverse is probably also true.

Since the available 4-bit ALU chips strongly influence DP-X organization including instruction format, the corollary is that a highly integrated custom missile processor probably would not have the same organization as DP-X for optimum utilization of a particular technology.

From designs already completed at Hughes using SOS-CMOS technology for a 16-bit missile processor and the layout study of the MCU with the newer I<sup>2</sup>L process, it can be extrapolated that either process can be used in mechanizing a fully integrated DP, with vast reduction in chip count.

Significant conclusions of the technology study are:

- Present LP-STTL could yield adequate DP
- MSI 4-bit ALU chips with 16 registers dictates DP-X organization
- No available MCU chips are completely suitable
- Lengthy pipelining cannot completely make up for low speed components
- LSI I<sup>2</sup>L (or SOS-CMOS) could make processor much smaller
- Optimum LSI processor design is most probably different from DP X
- Cell arrays can significantly reduce package count and cost

## SECTION IX

### COST ANALYSIS

#### 9.1 INTRODUCTION

In the course of the system design studies, numerous design decisions were made. In most of these decisions cost was a factor. In many of these cases, the cost implications were clear enough that no detailed cost analysis was attempted; in such cases engineering judgement and experience was sufficient to settle the issue. The more important of this kind of decision were:

- (1) The total digital processing load in the weapon should not be concentrated in a single processor (i. e., it should be distributed among several processors).
- (2) A digital bus rather than an analog harness should be used.
- (3) The bus should be a serial rather than parallel bus.

The subsystem analysis studies showed that a number of the subsystems had functions which could (and should) be performed digitally. Two examples studied in detail were the E/O seeker and the radio-metric area correlation (RAC) sensor. In both cases there is (or could be) a significant digital processing load. Trying to centralize such processing had one or both of the following effects: excessively high transmission rates on the bus; or unrealistically high throughput requirements for the DP. Moreover, putting these functions into a central DP did not significantly enhance the integration function. In cases like this, the decision was made, based on engineering judgement and experience, that well localized functions requiring high bus rates or high processor throughput should not be centralized. Thus, the DP design is basically a distributed processor system.

Having decentralized seeker and sensor processing, there remains a residue of tasks that appear appropriate to the DP integration role. These are the core functions, and either are directly concerned with integration or are basic to most or all of the weapon configurations. A specification has been made for the required performance for the DP digital processing system to perform the core function (with adequate growth). At this point, the question again comes up as to the best configuration to implement the requirements. That is, should the core function be distributed among two or more processors or should it be centralized? How should the DP be designed: monoprocessor or multiprocessor? In contrast to the case with the overall system, there are several configurations that appear practical and it is difficult to make a priori judgements about the relative cost. Most of the questions have to do with the digital processor(s) and not the bus structure. That is, most of the practical configurations have approximately the same bus requirements.

The question of distributed versus central processing has taken on a new interest lately because of the wide selection of microprocessors available. In fact, one of the major questions before the digital designer these days is how to effectively use the various microprocessors. The function performed by the microprocessor does not represent the majority of the complexity in the total digital processing system. Memory, I/O and bus circuitry account for most of it. Yet, the choice of the way one chooses to implement the CPU function can effect the whole system. For example, if one chooses to use an MOS microprocessor CPU, one is forced to some kind of distributed processing. Therefore, one way of examining the different configurations is by examining the different approaches to CPU implementation.

Today there are four major approaches to CPU implementation:

1. Discrete MSI chips
2. Bipolar LSI microprocessor chip sets
3. MOS microprocessors
4. Custom LSI designs.

The last choice would be essentially a custom microprocessor. An approach using a custom LSI design differs from approaches two and three in that in the latter case, the system is designed around an existing CPU. In the former, the CPU is designed to satisfy system requirements.

The cost analysis study focussed on the cost differences between several practical overall architectures which could satisfy the DP requirements. The basic question was central versus distributed. However, subsidiary questions were examined also: Monoprocessor versus multiprocessor, and MSI versus LSI. In the latter, the LSI refers to the CPU function. In all the cases considered, it refers to a microprocessor, whether bipolar or MOS, commercial or custom. Variations in bus design were not deemed important in deciding the above question. Therefore, the same bus design was used in all configurations studied.

Maintenance costs and reliability were not explicitly treated in the study. In all of the variations considered, we are dealing with essentially the same kind of components which are subjected to the same environment. Under these conditions, reliability is largely a function of the number of components involved. The number of components for each variation studied is given in the test. There does not appear to be any significant differences in maintenance costs for the different configurations studied. Each configuration has the same number of boxes and cables. While the number of circuit boards varies across the configuration, this would not seem to have a major effect on maintenance costs. Another factor, which probably has little significance overall, is related to changing system memory. In the distributed processor cases (with several memory locations) there is more labor involved in changing memory. It is not expected that memory will be changed enough to make this an important factor.

The study reported in the following pages considers

- Development costs
- Production costs
- Development risks
- Software generation

for six different architectural configurations.

Note that the production cost figures given in this report are lower than given in the interim report. This is because of a reevaluation of material costs, and in particular to extrapolating all costs to 1980. In the interim report only the integrated circuit costs were extrapolated.

## 9.2 SELECTION OF PROCESSOR TYPES FOR COST ANALYSIS

Perhaps the most basic tradeoff in the cost analysis is central versus distributed processing. To investigate this question adequately requires looking at several other tradeoffs: (1) monoprocessor versus multiprocessor; (2) commercially available designs versus custom designs; and (3) the use of LSI versus MSI. The possible permutations on these choices would lead to 16 different designs. However, some of the designs are not interesting, are impractical, or give redundant information. Six configurations were chosen for the study. Figure 68 shows a decision tree which illustrates the chosen variations.

The branch at point A in Figure 68 represents the primary tradeoff. The branch at the next level is between monoprocessors and multiprocessors. A multiprocessor is an interesting enough concept to deserve inclusion in the study. (A multiprocessor consists of two or more CPUs working against a common memory.) However, the prime reason for including it here is the existence of microprocessor designs that can be or must be used this way if they are to satisfy the DP requirements. For example, the multiprocessor branch at point C is made to accommodate MOS microprocessors. The only practical way to use these devices in the DP is in a multiprocessor. The corresponding branch at point B is made more because of the intrinsic interest of the multiprocessor approach. There are in existence high performance microprocessor designs which appear suitable for this approach. At the present time the designs are custom (not commercially available). One such design is the Hughes SOS C-MOS processor. Three of these in a multiprocessor design should be able to satisfy the DP requirements.

The branches at the next level, commercial versus custom, are required because commercial sources do not exist for some of the interesting variations. An example of this is the branch to custom at point 4. At the present time there does not exist a one or two chip microprocessor which can satisfy the DP requirements. Since it is quite likely to be beyond 1980 when such a design becomes commercially available, a custom design was chosen for the study.



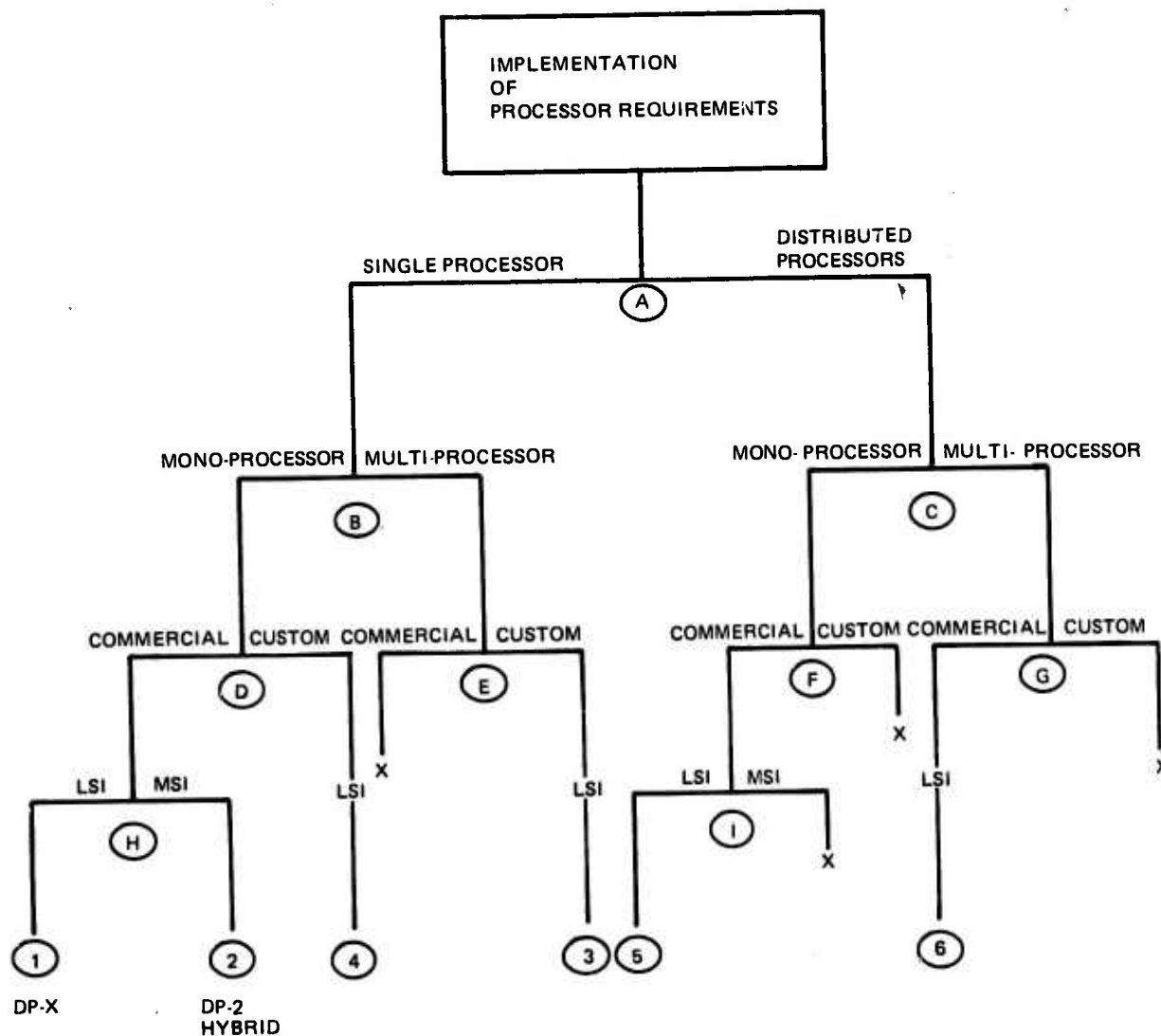


Figure 68. Selection of Processor Types for Cost Tradeoffs

The circled numbers at the bottom of Figure 68 identify the chosen variations. A brief description of each is given below and a more detailed description in the next section.

#### Configuration

#### Description

- |   |   |
|---|---|
| 1 | This design was generated in the technology study, where it was called DP-X. It uses commercial, bipolar microprocessors. |
| 2 | This is essentially the breadboard DP-2 packaged with hybrid integrated circuit modules.                                  |
| 3 | A multiprocessor using 3 custom design microprocessor CPUs.   |

- 4 A monoprocessor with a custom design microprocessor CPU.
- 5 Two distributed processors. This is very similar to the baseline design (Configuration 1) and also uses commercial, bipolar microprocessors. Each processor in this variation has somewhat more than one half of the throughput capability that variation one has.
- 6 Four distributed processors. Each processor is a multiprocessor with 4 MOS CPUs.

### 9.3 CONFIGURATION DESCRIPTIONS

#### 9.3.1 Overall Description

In this subsection the total digital processing system used in the cost analysis is described. While the major tradeoff areas have to do with processor implementation (and more specifically with CPU) it is necessary to include the entire system since changes in the processor may lead to other changes.

The DP weapon bus design allows for 16 stations including the master processor. However, it is not likely that all 16 will be used in most weapon configurations. For the purposes of the cost study, the weapon bus has eight stations. One of these is used by the DP (or the master DP for the distributed processor cases). Each of the other stations has a bus interface unit (BIU) or a satellite DP. Where a DP occupies the station (master or satellite) the BIU function is absorbed into the processor. The satellite processors also have to furnish an interface to the "distributed element" (subsystem) located at that station.

The eight stations are connected together by a bus consisting of 10 twisted, shielded pairs. The connection to the bus at each station is by a single connector.

The digital processing system does not contain its own power supply. Power of the correct wave form is supplied to the master DP via a power connector. Power is distributed to the BIU's along the bus. Satellite processors are supplied by the weapon system through their own power connectors.

Thus, the system considered in the cost study consists of 8 boxes and the weapon bus. Of the 8 boxes,  $(1+N)$  are processors and  $(7-N)$  are BIUs where  $N$  is the number of satellite processors.

#### 9.3.2 Box Descriptions

The processor box (whether for master or satellite processors) have three external connectors. One is for input power and one is for

the bus connection. The third connector is used to interface with a distributed element or as a test connector for laboratory testing.

The processor contains a mother board which supplied all the interboard wiring. The mother board supplies space for 1 to 3 CPU boards (depending on the configuration) 1 to 3 memory boards, 2 I/O boards and 2 spares.

The BIU box has two connectors, one for the bus and one for the distributed element. It contains a single circuit card and no mother board.

### 9.3.3 Circuit Card Descriptions

In order to eliminate extraneous variables from the cost study, a standard circuit board was chosen for all configurations and for all functions within a configuration. The standard chosen was a 3x4-inch ceramic, multilayer board. The standard packaging technique used was to purchase all integrated circuits or chips, place them in lead-less carriers and place the carriers on the circuit boards. The only variation from this procedure was in Configuration 2 in which the chips are assembled in hybrid integrated circuits in 1-5/8x1-1/4-inch module modules. This latter arrangement was chosen because Configuration 2 corresponds very closely to an existing design (packaged in hybrids) for which we have a considerable amount of information. As it turned out, this exception did lead to an anomaly in the study results which will be discussed later.

#### Central Processing Unit (CPU) Cards

The CPU design is the source of all the variations among the six configurations. The designs vary from MSI (Configuration 2) through bipolar microprocessors (Configurations 1 and 5), MOS microprocessors (Configuration 6) to custom LSI designs (Configurations 3 and 4). Configuration 1, the base line, is based upon the DP-X design developed in the technology task. It is designed around the AMD 2901 bit sliced, bipolar microprocessor. This design is highly pipelined in order to obtain the throughput required for DP. The design has two functionally identical parts each of which can fit on one circuit board. Each contains 68 integrated circuits. These boards are essentially arithmetic units (with their own microprogram control); one for address arithmetic and one for general arithmetic. Additional CPU functions are contained on a third board with 60 integrated circuits.

Configuration 5 is similar to Configuration 1 in that it uses bipolar microprocessors in the CPU. However, since it is a distributed configuration, each processor does not require as high a throughput as Configuration 1. Hence, the high degree of pipelining is not required and the design is much closer to the standard design using these microprocessors. The design can be obtained from Configuration 1 essentially by eliminating one of the arithmetic boards. Thus, the configuration 5 has two processors. Each processor has a CPU which requires

2 circuit boards. One circuit board is equivalent to one of the micro-programmed arithmetic boards of Configuration 1 and the other board contains additional CPU functions and is equivalent to the third CPU board in configuration one.

Configuration 2 is based upon the breadboard DP-2. The design uses MSI integrated circuits placed in hybrid modules. These hybrid modules are very similar to the modules used in the Hughes GMP design. Ten hybrid modules, 21 PROMs for the microdecode function, and 5 miscellaneous MSI chips are required for the CPU. These parts are placed on three circuit boards.

Configurations 3 and 4 are similar in that they both use custom LSI designs. Configuration 3 is a multiprocessor design. That is, it uses three CPUs but a single memory. Each processor in this design is similar to the Hughes SOS C-MOS processor and has a throughput somewhat greater than 1 million short operations per second. The technical problem in this design (excluding software considerations) is controlling access to the program and data memories to minimize interference among the processors. A preliminary design for the memory management and other ancillary CPU functions was partitioned for LSI cell arrays. Seven such cell arrays plus five additional MSI chips are required. Thus, this CPU requires 3 custom designed LSI chips, 7 LSI cell arrays and 5 MSI chips.

Configuration 4 is not a multiprocessor. The major CPU functions are implemented in two I<sup>2</sup>L LSI chips. Some ancillary functions are required and these are contained in two LSI cell arrays for a total of 4 chips. Both Configurations 3 and 4 require only one circuit board for the CPU.

Configuration 6 was included in the cost study to determine how MOS microprocessors would perform in the DP role. On the surface, the application of MOS microprocessors seems quite appealing since they can include essentially the entire CPU functions on a single chip. These microprocessors are available in 4-bit, 8-bit and 16-bit versions. Unlike most of the bipolar microprocessors, the 4-bit and 8-bit devices cannot easily be combined to make a 16-bit processor. Hence, the study considered only the 16-bit devices. The best of the 16-bit microprocessors have a throughput (for short instructions) of the order of 250,000 operations per second. Hence, it would take a minimum of twelve such processors to satisfy the DP requirements even if there were no difficulty with their long instructions such as multiply and divide. However, this minimum number assumes that the total processing load can be divided into parcels which just fit the capacity of the microprocessors. In general, this is not the case and there is significant inefficiency in applying multiple processors to a set of defined processing tasks. For the purposes of this cost study the total efficiency was set at 75 percent. That is, it takes 16 microprocessors to fill the DP requirements.

The 16 microprocessors can be arranged in a number of ways. One way would be to have 16 separate processors, each with its own

memory and its own defined tasks. This approach was rejected for three reasons:

- (1) It was inconsistent with the overall arrangement chosen for the study.
- (2) The total processing load cannot conveniently be broken up into 16 separate parts.
- (3) Even if each CPU and its memory could be combined upon one circuit card, the total number of circuit cards would exceed that for other (e. g., the selected) arrangements.

Somewhat arbitrarily it was elected to limit the configuration to four processors of identical design. Therefore, each processor includes 4 microprocessors in a multiprocessor design. As in Configuration 3, additional circuitry is required for memory management. The requirements for memory management and other ancillary functions can be approximated by extrapolating from the configuration 3 design. The latter required 7 LSI cell arrays and 5 MSI chips for these functions. Therefore, it is felt that ten LSI cell arrays and 6 miscellaneous chips will be adequate for this and other ancillary functions. Each CPU thus has 4 microprocessors, 10 cell arrays and 6 MSI chips all placed on 1 circuit card.

#### Memory Cards

The DP main memory is separated into two parts: the program memory and the data memory. The program memory uses ROM or PROM, while the data memory uses a combination of read/write memory (RAM) and ROM or PROM.

#### Program Memory

The baseline configuration (number 1) uses a 16-bit program memory word. For purposes of the cost study it was assumed that all configurations would use the same size word and that 8000 words are required. The DP breadboard (on which Configuration 2 was based) uses a 24-bit program memory word, however, with this size word a somewhat smaller number of words would be required; therefore, to to simplify the study, it was assumed that the total number of required bits would be the same.

To determine chip count, a standard chip of 512 words by 8 bits was assumed. Sixteen of these chips (making up 4K words) can be placed on the standard circuit card if they are packaged in the leadless carrier. In addition to the memory chips, 8 MSI chips are required for memory control and buffering. In the first four configurations, two of these memory cards (PM cards) are required to make up the total 8K words.

For the distributed processor configurations (5 and 6), additional program memory is required because of duplication of functions (e. g., initialization routines) and inefficiencies in distributing the functions

among the memory modules. The actual increase can only be determined from a detailed analysis for each possible case; however a feeling for the magnitude of the increase can be had from the following argument. Statistically, over a large number of systems, the end-point of the total program assigned to a processor will have a uniform distribution over the last memory module required. Therefore, the mean unused memory for each processor is one half of a memory module. Thus, each additional processor added to the system results in an additional one-half of a memory module on the average. This increase is in addition to any real increase in requirements due to duplication of functions. Thus, if the total processing load is split among two processors, the memory needed increases by something greater than one half of a memory module. If the load is split among four processors, the memory needed increases by more than one and one half memory modules. Since the memory, physically, comes in whole modules, the two cases cited above require an additional one and two memory modules, respectively.

Based on the above argument, Configuration 5 requires an additional 512 words (two chips) per system and Configuration 6 requires an additional 1024 words (4 chips) per system. In Configuration 5, each processor will require one memory board. The board will be the same design for both processors, but one will have an additional two memory chips on it. In Configuration 6, the program memory requirement for each processor is assumed to be small enough that it can be combined on one board with the data memory.

#### Data Memory

The total required address space for the DP in one of the first four configurations is 4K words. These words are allocated over scratch pad, constants, flags and I/O addresses. For the purposes of the cost study, the allocation shown in Table 23 was used. Clearly, a 256x1 RAM could have been used for the flags, however, this would have added another part number to the system.

The chips listed in the table, along with 8 MSI chips, for memory control and buffering, are placed on one of the standard circuit boards to form the data memory (DM) board for Configurations 1 through 4.

TABLE 23. ALLOCATION OF DATA MEMORY ADDRESS SPACE

NUMBER OF WORDS	USE	CHIPS
2560	16-BIT CONSTANTS	10 512 X 8 PROMS
1024	16-BIT SCRATCH PAD	16 256 X 4 RAMS
256	1-BIT FLAGS	1 256 X 4 RAM
256	I/O ADDRESSES	NONE REQUIRED



Just as in the case of program memory, the distributed processors require an additional amount of data memory. The allocations for Configuration 5 is shown in Table 24. Each processor in the configuration requires one circuit board for the data memory. Each circuit board requires 8 MSI chips for memory control and buffering.

In Configuration 6, the total memory requirement for each processor is small enough that the program memory and data memory can be combined on one circuit board. The memory arrangement for each processor is shown in Table 25.

In the first five configurations, a fast, bipolar memory is required. In Configuration 6, however, it is possible that a slower memory can be used. N MOS memories built to military requirements are available at significantly less cost than bipolar memories. N MOS memories with 300 nanosecond access time have historically been procured at about 80 percent of the cost of bipolar memories. N MOS memories with 500-nanosecond access time have been procured at about 40 percent of bipolar memory costs. In this cost study, it was assumed that the 300-nanosecond access time would be required to satisfy the needs of the multiprocessor configuration.

#### Input/Output (I/O) Circuit Cards

In the breadboard DP-2 system, the BIU function is external to the DP. Internal to the DP are I/O circuits to interface with the BIU, and circuits to perform the bus master control function. Considerable circuitry can be saved by combining these functions. This approach was taken for the cost study.

#### Single Processor Configurations

In the Configurations 1, 2, 3 and 4, the processor I/O circuits perform four functions:

- Processor Master clock
- Interface with BIU
- BIU function
- Bus master control function.

A preliminary design combining these four functions was made using a microprocessor. The microprocessor chosen was the SMS 300. The use of the microprocessor eliminates a significant amount of random logic circuitry in the present design but not all of it. There were three approaches for implementing the remaining random logic:

- Available MSI
- Custom LSI
- Semi-custom LSI.



TABLE 24. DATA MEMORY ALLOCATION FOR CONFIGURATION 5

TYPE	PROCESSOR NUMBER 1		PROCESSOR NUMBER 2	
	NUMBER OF WORDS	NUMBER OF CHIPS	NUMBER OF WORDS	NUMBER OF CHIPS
CONSTANTS	2000	8	1000	4
SCRATCH PAD	750	12	500	8
FLAGS	256	1	256	1

TABLE 25. MEMORY ARRANGEMENT FOR CONFIGURATION 6

MEMORY TYPE	PROCESSOR				TOTAL
	1	2	3	4	
PROGRAM	4K WORDS	2K WORDS	1.5K WORDS	1.5K WORDS	9K WORDS
	16 CHIPS	8 CHIPS	6 CHIPS	6 CHIPS	36 CHIPS
CONSTANTS	1.5K WORDS	0.5K WORDS	0.5K WORDS	0.5K WORDS	3.0K WORDS
	6 CHIPS	2 CHIPS	2 CHIPS	2 CHIPS	12 CHIPS
SCRATCH PAD	0.5K WORDS	0.5K WORDS	0.25K WORDS	0.25K WORDS	1.5K WORDS
	8 CHIPS	8 CHIPS	4 CHIPS	4 CHIPS	24 CHIPS
FLAGS	256 WORDS	256 WORDS	256 WORDS	256 WORDS	--
	1 CHIP	1 CHIP	1 CHIP	1 CHIP	4 CHIPS
CONTROL AND BUFFERING	12 CHIPS	12 CHIPS	12 CHIPS	12 CHIPS	48 CHIPS
TOTAL	43 CHIPS	31 CHIPS	25 CHIPS	25 CHIPS	124 CHIPS

The last technique was chosen for the cost study. In particular, an LSI cell array approach, proposed by Hughes Newport Beach, was used. This approach combines six to eight MSI cells in a single LSI chip. Development time for an LSI array is about 12 weeks and studies have shown that savings relative to separate MSI chips are significant. The technology chosen was low power Schottky. The remaining random logic was partitioned into 8 LSI cell arrays. The total parts count for the I/O circuitry is given in Table 26. Two circuit boards will accommodate these parts.

#### Distributed Processor Configurations

In these configurations, the master DP must perform the same I/O functions the single DP does in the preceding configurations. Therefore it requires the same I/O circuitry. The satellite DPs could be connected into the system by means of an external BIU, or the BIU could

TABLE 26. PARTS LIST FOR PROCESSOR INPUT/OUTPUT

PART	NUMBER USED
SMS 300 MICROPROCESSOR	1
INTERFACE VECTOR BYTES (SMS)	9
512 X 8 PROMS	2
SMALL PROMS	3
4-BIT X 16 WORD FIFO	16
LSI CELL ARRAYS	8 (8 TYPES)
QUARTZ CRYSTAL	1

be absorbed into the satellite DP I/O. The overall system cost is probably less for the latter arrangement. The functions to be performed by the satellite processor I/O are:

- Processor master clock
- Interface with BIU
- BIU function
- Interface with one or more weapon subsystems.

A detailed analysis of the circuitry differences between the master and satellite I/O's was not done. Instead, the assumption was made that the two sets of functions would require equivalent amounts of circuitry. Since the functions are not identical, however, at least one of the circuit boards in the satellite DP I/O will differ from the corresponding board in the master DP I/O. Thus, in the complete system, there are three I/O circuit board designs. The master DP uses one each of designs one and two, and each satellite DP uses one each of designs one and three. The cost of designs two and three is assumed to be the same.

#### Bus Interface Unit (BIU) Card

The LSI cell array approach was also used for the BIU design treated in the cost study. The circuitry for the data slave function, the interrupt function and the bus race function was partitioned into nine cell arrays. In addition, three small PROMs are required. This circuitry fits on one standard circuit card in the BIU box.

#### 9.4 Production Costs

The estimated production costs for the six different configurations were developed using a computer-based cost model. The cost model is based on experience from numerous programs which have had equipment produced in Hughes Tucson factory, and has demonstrated good accuracy in predicting production costs. Outputs from any cost

model are only as good as the inputs. The major inputs to the cost model are the prime material costs and the labor costs. This section describes the cost model and the basis for material and labor cost estimates.

#### 9.4.1 Cost Model

The cost model summarizes costs at various levels. The three top levels are selling price, manufacturing cost and factory cost. The relations between these costs are defined by the following equations:

$$\text{Price} = C_M (1 + \alpha)(1 + \beta)$$

$$C_M = C_F (1 + S_F + S_E)$$

The following definitions apply:

Price - The cost to the customer.

$C_M$  - Cost at manufacturing level or manufacturing cost.

$\alpha$  - General and administrative cost factor (G&A).

$\beta$  - Profit factor.

$C_F$  - Factory cost.

$S_F$  - Factory support factor. This factor takes into account the effort required to maintain the production capability and solve manufacturing problems. The factor is expressed as a percentage of the factory cost. In general, it will vary with the type of program. Introduction of new manufacturing methods would increase this factor, for example.

$S_E$  - Engineering support factor. This factor takes into account the support given by the engineering staff to the manufacturing process. It involves such things as design changes to accommodate production practices and ensuring that produced equipment satisfies the specifications.

In the cost study reported here, the summary level used for comparison is the factory cost,  $C_F$ . The other factors used to obtain the higher summary levels are insensitive to the differences in the configuration and hence only tend to obscure the differences.

#### 9.4.2 Development of Factory Cost

Factory cost is the summation of two quantities: labor cost and material cost. Both of these latter costs involve a complex of input data.

The material costs start with the cost for supplying the basic or "prime" material. This is the cost of buying just enough material for supplying the needs of the system to be built. The prime material cost is adjusted by three factors: freight allowance, material attrition and material expense.

Freight allowance covers the expense of shipping the material and is expressed as a percentage of material costs. Obviously a straight percentage of material costs is inaccurate for this factor, but since the percentage is small, it does not lead to significant inaccuracies in the final result.

Material attrition allows for the fact that some of the material will fail and have to be scrapped and that the project will end with some surplus of material that has to be disposed of. The surplus may come about because of spec changes which require a new part. This factor is derived from historical data.

Material expense covers the indirect effort required to order and stock the material. The factor is derived from historical data.

The equation which describes how the above three factors operate is:

$$C_{MT} = C_{PM} (1 + A_F + A_A)(1 + E_M).$$

The pertinent definitions are:

$C_{MT}$  = Material cost

$C_{PM}$  = Cost for the prime material

$A_F$  = Freight allowance

$A_A$  = Allowance for attrition

$E_M$  = Material expense

With respect to the present cost study, the only variable is the cost for prime material. These costs will be discussed in a following section.

Labor costs are developed from four factors; standard hours, labor index, labor rate and labor expense. The formula connecting these quantities is

$$C_L = (H_{STD})(\text{index})(\text{rate})(E_L).$$

In the above equation, the following definitions apply.

$C_L$  - Labor cost

$H_{STD}$  - Standard hours

Index - Labor index which adjusts standard hours to actual hours

Rate - Hourly (or monthly) rate for the particular labor involved (fab, assembly or test).

$E_L$  - Labor expense or overhead.

There are four types of labor for which standard hours are estimated. These are fabrication, assembly, inspection, and test. These estimates are computed on the basis of time it would take a trained worker to accomplish the task if there were no set-up time, rework, or stops for inspection. For well established procedures, the standard hours are estimated on the basis of experience. For new procedures, a time and motion study is done.

The labor index adjusts standard hours to actual hours. It takes into account the effort required for set-up for the task, rework required and inefficiencies due to such things as waiting for an inspector to approve the work.

There are separate labor rates for each category of labor. These rates are established on the basis of negotiation or history from similar on-going projects. Labor expense is a negotiated quantity. For the present study, 1976 labor rates and expense were used.

#### 9.4.3 Prime Material Costs

Material costs were gathered on the basis of requirements for 1000 systems. Extrapolations of costs for a larger number of systems is done in the computer cost model. Most of the costs were based on vendor planning purpose quotes. Where the quote did not cover the necessary quantity or time period, extrapolations were made. The assumption was made that material would be purchased in 1980. Extrapolations to that time period included effects of technological advance and competition but did not include inflation. Prices are in 1976 dollars.

#### Costs for Integrated Circuits

The costs of ICs used in the cost analysis are shown in Table 27. The origins of the costs are discussed below.

#### Memory

The costs of semi-conductor memories have declined rapidly over the last several years. Figure 69 illustrates the trend for military quality memories. By 1980, RAM should be available for about 1.5¢ per bit and PROM for a few tenths of a cent per bit. The chart indicates that the price reduction is accompanied by (and to a large part a result of) a corresponding increase in density (bits/chip). For the purposes of the study, the price was extrapolated to 1980 but chip densities corresponding to 1976 were used. This inconsistency might affect the study results in two ways. First, the higher density chips expected in the future will allow packaging on less board space, leading to a cost reduction, particularly in Configurations 1 through 5. Second, the memory module size will probably go up. For example, module size for program memory might be most economical at 1K or 2K words rather than 500 words. This effect would affect primarily Configurations 5 and 6 (the distributed processor cases). Total memory usage will increase in those cases.

TABLE 27. ESTIMATED COSTS FOR INTEGRATED CIRCUITS IN 1980

TYPE	COST (DOLLARS)		
	1 TO 4	CONFIGURATION 5	6
512 X 8 ROM	8.75	8.75	7.00
SMALL ROM	3.00	3.00	3.00
1K RAM	15.62	15.62	12.50
FIFO	6.20	5.00	3.00
SOS C-MOS CPU CHIP	50.00	N/A	N/A
i <sup>2</sup> L CPU CHIP	50.00	N/A	N/A
MOS CPU	N/A	N/A	25.00
AMD 2901	22.85	22.85	N/A
AMD 2914	10.00	9.00	N/A
SMS 300	40.00	37.50	35.00
IV BYTE	7.00	6.50	6.00
CELL ARRAYS	5.30	5.30	4.93
HYBRIDS	125.00	N/A	N/A
OTHER MISCELLANEOUS MSI CHIPS	1.00	1.00	1.00
QUARTZ CRYSTAL	10.00	9.00	8.00

Configuration 6 uses N MOS memories which is cheaper than bipolar memories. Experience has been that military grade N MOS memories with 300 nanosecond access time can be procured at about 80 percent of the corresponding bipolar types. The 500-nanosecond memories are less, about 40 percent of the bipolar. Since the memory access time requirements of the 4-processor multiprocessor has not been analyzed in detail, the conservative approach of specifying 300 nanosecond memory has been taken.

The estimated costs for the FIFOs (first in-first out) memories is based on a vendor quote.

#### Custom LSI Designs

A cost projection, based on past experience, was made to give a basis for the cost estimates for the custom design CPUs in Configurations 3 and 4. The ground rules for the projection were:

- Functional complexity and speed requirements in the GMP range.

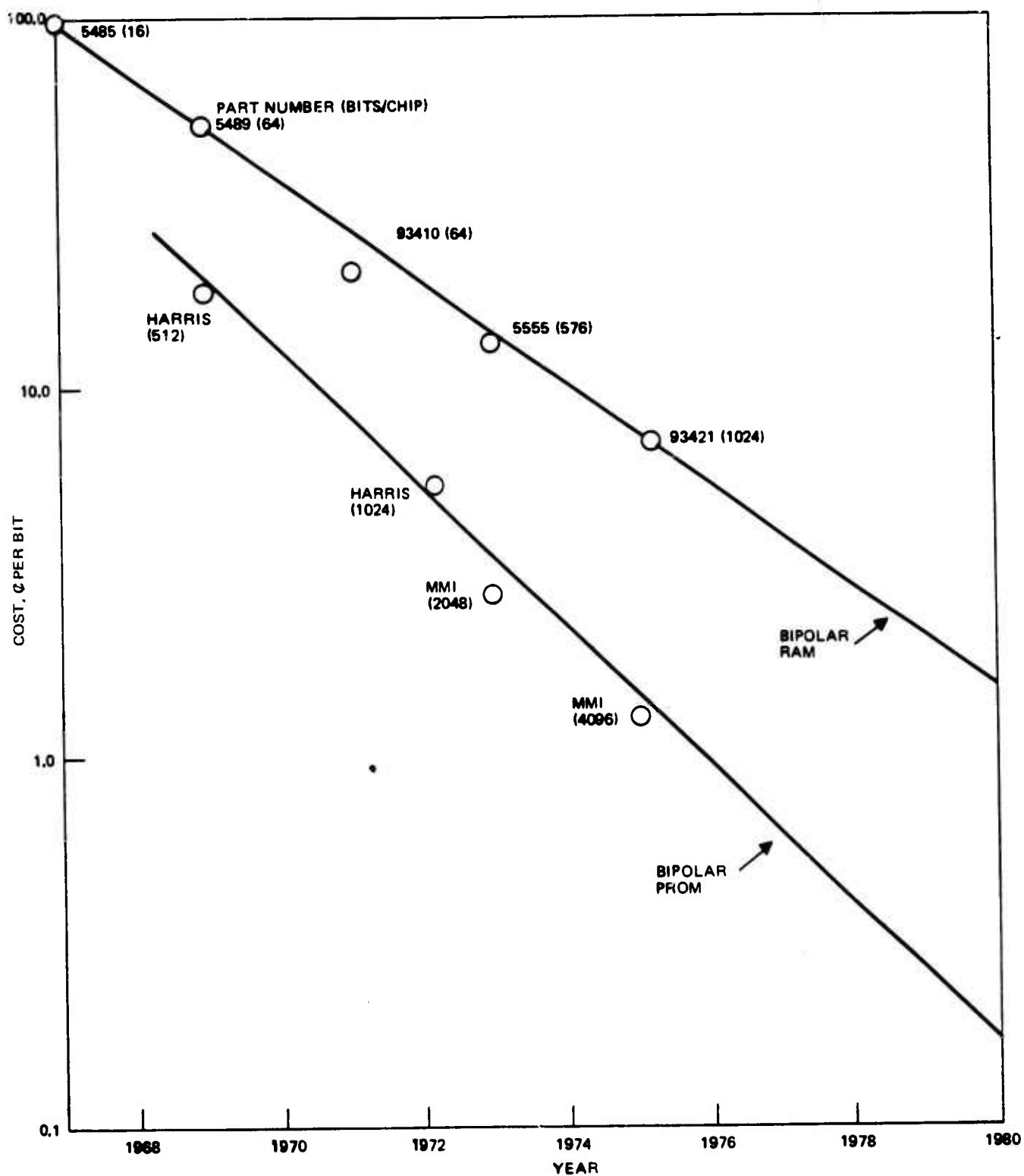


Figure 69. Cost History for Military Memories



- Full burn-in and screening required
- Prototype design in 1978 with production units procured in 1980
- Costs for 1000 units
- Chip size will be limited to about 0.2x0.2-inch.

SOS C-MOS and  $I^2L$  were considered to be the most promising candidate technologies. The results of the projection are shown in Table 28. The use of CMOS on sapphire requires only moderate extension of current technology. Achievement of the required functional complexity may cause the chip size to exceed 0.2x0.2-inch or may require a CPU set comprised of two chips as shown in the table. The use of IIL technology is based upon projections that new improvements in this technology will emerge more rapidly than for the other candidate technologies. Higher speed nonsaturating configurations of IIL devices have already been developed. Higher packaging densities of IIL and inherent flexibilities make it probable that the CPU can be implemented in one chip. The complexity and speed requirements of the custom chips are great enough to push the state of the art. Hence chip yield will be low as shown in the table.

Considering the results of the above cost projection, a more conservative approach was taken for the cost analysis. In the SOS C-MOS case, it was decided to use a chip of somewhat less complexity in a multi-processor configuration. The model is the Hughes SOS C-MOS single-chip processor. Three of these are required for the multi-processor of Configuration 3. Because the complexity is less, the yield will be higher and cost lower (per chip) than shown in Table 27.

For the  $I^2L$  case, the more conservative approach of using a two chip design was chosen. Again the complexity of each chip is less, yield is higher and cost is down from the results shown in Table 27.

TABLE 28. PROJECTED COSTS FOR LSI MONOPROCESSOR CPU

ITEM	SOS CMOS	$I^2L$
• WAFER PROCESSING COST	\$ 65.00	\$ 65.00
• DEVICE YIELD PER WAFER	4	3
• PROCESSING COST PER DEVICE	16.25	21.66
• ASSEMBLY, TEST, PACKAGING	8.00	9.00
• PACKAGE YIELD	75 percent	75 percent
• MANUFACTURING MARGIN	65 percent	65 percent
• SELLING PRICE PER CHIP	54.00	87.50
• NUMBER OF CHIPS REQUIRED PER SET	2	1
• SELLING PRICE PER SET	\$ 108.00	\$ 87.50

It was estimated that each chip would cost about \$50, yielding a total cost higher than shown in the table.

#### Other LSI

Most of the other LSI prices were based on vendor quotes. In some cases the vendor extrapolated the quotes to 1980. In the other cases it was assumed that prices would drop 10 to 15 percent per year, a decline which has been typical for such parts. The prices given for the MOS CPU and the 2914 are estimates made without benefit of a vendor quote. The system costs for integrated circuits for each configuration are given in tables 29 through 35.

#### Other Material Costs

The costs for the other material used in the system is summarized in Table 35 with an indication as to the source of the data. Costs for connectors were estimated as were costs for mother board material and miscellaneous hardware. The miscellaneous electronics includes resistors and capacitors for bypassing. These were counted at one resistor or capacitor for every four integrated circuits. Table 36 also summarizes the total material cost for each system.

#### 9.4.4 Labor Costs

There is only one aspect in which the labor costs for the six configurations might differ from historical costs. This is in the process of putting the integrated circuit chips into the leadless carriers. This is a new process for the Tucson factory and required a time and motion study to estimate the labor. All of the configurations used this process in all or part of the equipment. The only configuration that did not use it for everything was 2, which had the CPU in hybrid integrated circuit modules. Since the costs for the hybrid modules were based on the estimates of a different Hughes factory (which has experience in producing the modules) and the cost of the use of the leadless carriers was based on a study, it may be expected that Configuration 2 will be atypical relative to the other configurations. The labor costs for using the leadless carriers were estimated on a conservative basis since it is a new process. It is felt therefore, that all the other configurations are priced high relative to Configuration 2.

#### 9.4.5 Production Cost Comparisons

The production cost estimates (at factory cost level) are shown in Table 35. The table shows the cumulative average cost per unit for producing 5000 units. While the total variation from the mean is only about  $\pm 20$  percent, it is felt that this difference is large enough to show trends and give significant information about the major tradeoffs.

#### Bipolar, LSI Microprocessors Versus MSI

This tradeoff is illustrated by Configurations 1 and 2. The former uses microprocessors. As seen from the table, there is not any significant differences in the cost. Configuration 1 would have had fewer

TABLE 29. NUMBER OF CHIPS OF EACH TYPE USED IN CONFIGURATION 1

TYPE OF CIRCUIT CARD	NUMBER OF CARDS/SYSTEM				PROMS								READ WRITE				LSI DEVICES										MISC. MSI AND QUARTZ CRYSTALS	TOTAL CHIPS	
	PROGRAM 512 X 8	CONSTANTS 512 X 8	MICRO PROGRAM 512 X 8	512 X 6	SMS 300, MICRO PROG. 512 X 8	MISC. 512 X 8	SMALL PROMS 256 X 4 RAMS	SCRATCH PAD 16 X 4	FIFOS	SOS C-MOS CPU CHIP	1-L CPU CHIP	MOS MICRO- PROCESSOR	AMD 2901	AMD 2914	SMS 300 MICRO PROC.	SMS IV BYTE	CELL ARRAYS	HYBRID MODULES											
CPU 1	2												8														116	136	
CPU 2	1												1														59	60	
CPU 3	0																												
PROG. MEM 1	2	32																									16	48	
PROG. MEM 2	0																												
DATA MEM 1	1	10																									8	35	
DATA MEM 2	0																												
PM/DM 1	0																												
PM/DM 2	0																												
PM/DM 3	0																												
I/O 1	1				2	3									1	9	4										1	20	
I/O 2	1							16									4											20	
I/O 3	0																												
8IU	7					21											63											84	
TOTAL CHIPS OR CARDS	15	32	10	12	2	24	17	16	0	0	0	8	1	1	9	71										200	403		
COST/CHIP - DOLLARS																													
COST/COLUMN - DOLLARS	280	88	105	18	72	266	99	0	0	0	0	183	10	40	63	376										209			
SYSTEM COST - DOLLARS																												1809	

TABLE 30. NUMBER OF CHIPS OF EACH TYPE USED IN CONFIGURATION 2

TYPE OF CIRCUIT CARD	PROMS										LSI DEVICES										TOTAL CHIPS	
	NUMBER OF CARDS/SYSTEM	PROGRAM 512 X 8	CONSTANTS 512 X 8	MICRO PROGRAM 512 X 8	SMS 300, MICRO PROG. 512 X 8	MISC. 512 X 8	SMALL PROMS 256 X 4 RAMS	SCRATCH PAD 16 X 4	FIFOs	SOS C-MOS CPU CHIP	1-2 CPU CHIP	MOS MICRO- PROCESSOR	AMD 2901	AMD 2914	SMS 300	MICRO PROC.	SMS IV	BYTE	CELL ARRAYS	HYBRID MODULES		MISC. MSI AND QUARTZ CRYSTALS
CPU1	1																		4		4	
CPU 2	1																		4		4	
CPU 3	1																		2	5	28	
PROG. MEM 1	2	32																		16	48	
PROG. MEM 2	0																					
DATA MEM 1	1		10																	8	35	
DATA MEM 2	0																					
PM/DM 1	0																					
PM/DM 2	0																					
PM/DM 3	0																					
1/0 1	1			2	3										1	9	4		4		1	20
1/0 2	1										16						4				20	
1/0 3	0																					
BIU	7											21						63				84
TOTAL CHIPS OR CARDS	15	32	10	2	45	17	16	16	0	0	0	0	0	0	1	9	71	10	30			243
COST/CHIP-DOLLARS																						
COST/COLUMN-DOLLARS		280	88	0	18	135	266	99	0	0	0	0	0	0	40	63	376	1250	39			
SYSTEM COST-DOLLARS																						2654

TABLE 31. NUMBER OF CHIPS OF EACH TYPE USED IN CONFIGURATION 3

TYPE OF CIRCUIT CARD	PROMS										READ WRITE		LSI DEVICES										QUARTZ CRYSTALS	TC TAL CHIPS
	NUMBER OF CARDS/SYSTEM	PROGRAM 512 X 8	CONSTANTS 512 X 8	MICRO PROGRAM 512 X 8	SMS 300, MICRO PROG. 512 X 8	MISC. 512 X 8	SMALL PROMS	256 X 4 RAMS	SCRATCH PAD	16 X 4 FIFOS	SOS C-MOS CPU CHIP	1/2 CPU CHIP	MOS MICRO- PROCESSOR	AMD 2901	AMD 2914	SMS 300 MICRO PROC.	SMS IV BYTE	CELL ARRAYS	HYBRID MODULES	MISC. MSI AND MODULES				
CPU 1	1										3							1			5	15		
CPU 2	0																							
CPU 3	0																				16	48		
PROG. MEM 1	2	32																						
PROG. MEM 2	0																							
DATA MEM 1	1		10																		8	35		
DATA MEM 2	0																							
PM/DM 1	0																							
PM/DM 2	0																							
PM/DM 3	0																							
I/O 1	1				2	3										1	9	4			1	20		
I/O 2	1								16									4				20		
I/O 3	0																							
BIU	7					21												63				84		
TOTAL CHIPS OR CARDS	13	32	10	0	2	24	17	16	3	0	0	0	0	0	0	1	9	78	0	30		222		
COST/CHIP - DOLLARS																								
COST/COLUMN - DOLLARS	280	88	0	18	72	266	99	150	0	0	0	0	0	0	0	40	63	413	0	39				
SYSTEM COST - DOLLARS																						1528		

TABLE 32. NUMBER OF CHIPS OF EACH TYPE USED IN CONFIGURATION 4

TYPE OF CIRCUIT CARD	PROMS										LSI DEVICES										QUARTZ CRYSTALS	TOTAL CHIPS		
	NUMBER OF CARDS/SYSTEM	PROGRAM 512 X 8	CONSTANTS 512 X 8	MICRO PROGRAM 512 X 8	SMS 300, MICRO PROG. 512 X 8	512 X 8 MISC.	SMALL PROMS 256 X 4 RAMS	SCRATCH PAD 16 X 4	FIFOS	SOS C-MOS CPU CHIP	1/2 CPU CHIP	MOS MICRO- PROCESSOR	AMD 2901	AMD 2914	SMS 300 MICRO PROG.	SMS IV BYTE	CELL ARRAYS	HYBRID MODULES	MISC. MSI AND MODULES					
CPU 1	1										2										2			4
CPU 2	0																							
CPU 3	0																							
PROG. MEM 1	2	32																						
PROG. MEM 2	0																							
DATA MEM 1	1		10																					
DATA MEM 2	0																							
PM/DM 1	0																							
PM/DM 2	0																							
PM/DM 3	0																							
I/O 1	1				2	3										1	9	4						20
I/O 2	1																	4						20
I/O 3	0																							
BIU	7						21											63						84
TOTAL CHIPS OR CARDS	13	32	10	0	2	24	17	16	0	2	0	0	0	0	1	9	73	0	25	211				
COST/CHIP - DOLLARS																								
COST/COLUMN - DOLLARS		280	88	0	18	72	266	99	0	100	0	0	0	0	40	63	387	0	34					
SYSTEM COST - DOLLARS																								1447



TABLE 33. NUMBER OF CHIPS OF EACH TYPE USED IN CONFIGURATION 5

[illegible]



TABLE 34. NUMBER OF CHIPS OF EACH TYPE USED IN CONFIGURATION 6

TYPE OF CIRCUIT CARD	PROMS										LSI DEVICES										TOTAL CHIPS	
	NUMBER OF CARDS/SYSTEM	PROGRAM 512 X 8	CONSTANTS 512 X 8	MICRO PROGRAM 512 X 8	SMS 300, MICRO PROG. 512 X 8	MISC. 512 X 8	SMALL PROMS	256 X 4 RAMS	SCRATCH PAD	16 X 4 FIFOS	SOS C-MOS CPU CHIP	1-L CPU CHIP	MOS MICRO- PROCESSOR	AMD 2901	AMD 2914	SMS 300 MICRO PROC.	SMS IV BYTE	CELL ARRAYS	HYBRID MODULES	MISC. MSI AND QUARTZ CRYSTALS		
CPU 1	4																				24	80
CPU 2	0																					
CPU 3	0																					
PROG. MEM 1	0																					
PROG. MEM 2	0																					
DATA MEM 1	0																					
DATA MEM 2	0																					
PM/DM 1	1	16	6																		12	43
PM/DM 2	1	8	2																		12	31
PM/DM 3	2	12	4																		24	50
I/O 1	4				8	12										4	36	16			4	80
I/O 2	1								16								4					20
I/O 3	3								48									12				60
BIU	4					12												36				48
TOTAL CHIPS OR CARDS	20	36	12	0	8	24	28	64	0	0	0	16	0	0	0	4	36	108	0	76		412
COST/CHIP - DOLLARS																						
COST/COLUMN - DOLLARS		252	84	0	56	72	350	192	0	0	0	400	0	0	0	140	216	532	0	104		
SYSTEM COST - DOLLARS																						2398

TABLE 35. TOTAL MATERIAL COSTS

ITEM	CONFIGURATION						SOURCE OF COST DATA
	1	2	3	4	5	6	
INTEGRATED CIRCUITS AND HYBRID INTEGRATED CIRCUIT MODULES	1,809	2,654	1,528	1,447	2,166	2,398	SEE TABLES 26 AND 28 THROUGH 33.
CHIP CARRIERS	402	232	221	210	506	408	VENDOR QUOTE
CIRCUIT BOARDS	639	649	549	549	778	826	VENDOR QUOTE
CIRCUIT BOARD CONNECTORS	300	300	260	260	360	400	ESTIMATE
MOTHER BOARD CONNECTORS	20	20	20	20	40	80	ESTIMATE
BOX CONNECTORS	340	340	340	340	360	400	ESTIMATE
HARNESS	992	992	992	992	992	992	VENDOR QUOTE
MOTHER BOARD MATERIAL	102	102	82	82	124	168	ESTIMATE
MISCELLANEOUS ELECTRONICS; RESISTORS, ETC.	100	60	55	52	127	103	ESTIMATE
SHEET METAL, SCREWS, MISCELLANEOUS MATERIAL	75	75	75	75	75	75	ESTIMATE
TOTAL	4,779	5,424	4,122	4,027	5,528	5,850	

TABLE 36. CUMULATIVE AVERAGE COST PER UNIT FOR 5000 UNITS

	CONFIGURATION					
	1	2	3	4	5	6
MATERIAL COST PER UNIT-DOLLARS	5,035	5,714	4,342	4,242	5,824	6,163
LABOR COST PER UNIT-DOLLARS	2,279	1,524	1,435	1,388	2,865	2,546
FACTORY COST PER UNIT-DOLLARS	7,314	7,238	5,777	5,630	8,689	8,709
DEVIATION FROM MEAN-PERCENT	1.2	0.1	-20	-22	20	20

parts, and therefore lower cost, if a suitable LSI microprogram control chip had been available for the design. Furthermore, as discussed previously, Configuration 2 is probably priced low in comparison with the other designs. If the use of an LSI MCU chip could have eliminated 40 MSI chips, the burdened material costs for configuration 1 would drop by perhaps \$300. If the labor for Configuration 2 were raised to equal labor for Configuration 1, its cost would increase about 700 dollars. For these two adjustments, the cost of Configuration 1 relative to Configuration 2 would drop \$1000. The new costs would then be approximately \$7000 and \$7900. This is a great enough difference to show a favorable trend for the use of bipolar microprocessors, relative to MSI.

#### Commercial Microprocessor Versus Custom LSI

Configuration 1, 5, and 6 use microprocessors in the CPU. Configuration 3 and 4 use custom LSI. However, of these, only 1 and 4 compare monoproducts. Configuration 1, even if we deduct the \$300 for using an LSI MCU, is still about \$1400 greater than 4. This would appear to be a clear trend in favor of custom LSI.

#### Central Processor Versus Distributed Processors

The first four configurations are central and the last two are distributed designs. The trend in favor of the central designs is clear. A particularly good comparison is between 1 and 5, since they both use the same technology. Configuration 5 is a significant \$1300 greater than 1.

This is not an unexpected result. It has historically been observed that computing power of a system is proportional to the square of the cost (as long as one stays within the same technology). Thus, increasing the cost of a computing system by 40 percent should allow doubling of the computing power. This assumes a single processor, of course. Doubling the number of processors will double the computing power also, but at twice the cost. Thus, from the viewpoint of cost, one should have better results by designing a central processor system as long as the capability of the technology is not exceeded.

#### Monoprocessor Versus Multiprocessor

Configurations 3 and 4 are a multiprocessor and monoprocessor, respectively. The cost difference does not seem to be significant.

#### Use of Microprocessors

Commercial microprocessors are used in the CPUs of Configurations 1, 5, and 6. As discussed before, the bipolar microprocessor sets seem to compare favorably with MSI implementations (assuming that a suitable MCU chip becomes available). It compares unfavorably with a custom LSI design. It is clear that present day MOS microprocessors (Configuration 6) will not lead to an inexpensive system. These processors would have to have a very large increase in their processing capability (a factor of 3 or 4) before they could approach

the cost/performance figure for a custom LSI approach. In fact, close scrutiny of the material costs used in this study indicates that a lower bound on the costs for a commercial microprocessor system is given by Configurations 3 and 4. The latter two designs are microprocessor designs also, of course. The difference is that these are custom designed microprocessors, designed to accomplish the DP kind of job. The MOS processors are not.

## 9.5 DEVELOPMENT COSTS

This section derives estimates for the development costs of the six different configurations and makes some qualitative judgements relative to development risk. Only the hardware development costs are considered here. However, some general observations about software development are made.

The costs are given for a 30-month engineering development phase in which one breadboard, two development and 20 prototype units are built and tested.

### 9.5.1 Development Cost Model

The computer-based cost model used for development costs was derived from an analysis of historical data. It has proved quite accurate in predicting development costs.

The cost factors considered in the cost model are shown in Table 37, along with the total program summary for Configuration 1. Special ground rules for input to the cost model are listed below.

- (1) The program is assumed to be conducted and organized in a manner similar to the AIM-VAL engineering development program. This similarity pertains particularly to the interface between the systems engineering and hardware areas.
- (2) Costs for program management, systems engineering, and systems test and data are based on similarity to the AIM-VAL program. Cost variation for these items from configuration to configuration is less than the tolerance on the numbers for any given configuration. Hence, these costs are taken to be the same for all configurations.
- (3) Peculiar support equipment and training costs were not estimated.
- (4) No costs are included for a preproduction program.

The hardware development costs for the six configurations are given in Table 38. The backup data for the results shown in Table 37 are given in Tables 39 and 40. With the exception of Configuration 5, the development costs are closely grouped (about  $\pm 7$  percent if Configuration 5 is not included). The variation with respect to the total program costs are even smaller. (Costs for program management, systems engineering, and system test and data must be added to the numbers given in Table 37 to obtain total program costs.) It is doubtful if such variations are significant. The rather high cost for

TABLE 37. PROGRAM COST SUMMARY FOR CONFIGURATION 1

SOURCE OF DATA: NOTED ON COVER LETTER; DATE: 20 MAY 1976 (COSTS IN 1975 K\$ AT MCL)						
	LABOR		ODC		TOTAL	
	HOURS	percent	K\$	percent	K\$	percent
PROGRAM MANAGEMENT					872	12.8
SYSTEMS ENGINEERING AND ANALYSIS					714	10.5
SYSTEMS TEST AND EVALUATION					907	13.3
HARDWARE DEVELOPMENT					4,277	62.8
COMMAND AND LAUNCH EQUIPMENT					0	0
O AND M SUPPORT EQUIPMENT					0	0
TRAINING					0	0
DATA					43	0.6
FACILITIES AND EQUIPMENT					0	0
OTHER					0	0
TOTAL					6,813	100.0

TABLE 38. HARDWARE DEVELOPMENT COST SUMMARY  
(IN \$1000)

	CONFIGURATION					
	1	2	3	4	5	6
DEVELOPMENT LABOR	2955.4	3135.7	2445.2	2408.3	3528.8	2828.4
MANUFACTURING LABOR	673.7	711.4	519.2	511.3	824.3	701.5
TOTAL LABOR	3629.1	3847.1	2964.4	2919.6	4353.1	3529.9
ODC	647.3	690.6	739.3	802.5	746.9	782.7
COST PER UNIT	97.7	103.2	81.3	79.7	121.6	100.0
TOTAL NON-RECURRING	2321.6	2474.3	2078.6	2127.6	2667.3	2313.0
TOTAL RECURRING	1954.8	2063.4	1625.1	1594.4	2432.7	1999.6
TOTAL HARDWARE DEVELOPMENT COST	4276.5	4537.7	3703.7	3722.1	5100.0	4312.6
VARIATION FROM MEAN (percent)	0.02	6	-13	-13	19	0.8

TABLE 39. INPUT DATA FOR ESTIMATING HARDWARE DEVELOPMENT COSTS FOR CONFIGURATIONS 1, 2 AND 3

(DATE 19 MAY 1976)	CONFIGURATION					
	1		2		3	
	EXPECTED	COMMENTS/RISK	EXPECTED	COMMENTS/RISK	EXPECTED	COMMENTS/RISK
SPECIFICATIONS	61	2 UNIT, 1 HARNESS, 2 BIU 8 CARDS X 2 - 16 + 40 IC'S	63	2 UNIT, 1 HARNESS, 2 BIU 9 CARDS X 2 - 18 + 40 IC'S	49	2 UNIT, 1 HARNESS, 2 BIU 2 X 7 CARDS - 14, 30 IC'S
ANALOG STAGES	0		0		0	
DIGITAL COMPONENTS	239	12 + 227	137	125 + 12	126	114 + 12
BREADBOARD ASSEMBLIES	8	403 X 2/100	9.1	(233 + 220) X 2/100	4.4	222 X 2/100
EXPERIMENTAL DRAWINGS	94		102		84	
PRODUCTION DRAWINGS	94		102		84	
PARTS TO BE QUALIFIED	40		40		30	
NEW MATERIAL AND PROCESSES	2		2		2	
DEVELOPMENT TESTS	672		696		592	
ACCEPTANCE TESTS	1620		1620		1380	
BAYS OF STE	4		4		4	
DEVELOPMENT ASSEMBLIES	34		28		26.2	
PROTOTYPE ASSEMBLIES	340		280		262	
SUBCONTRACT SUPPORT, mm	21		22		23	
TOTAL COMPONENTS	500		566	(233 + 220) (1.25)	278	222 X 1.25
COMP. IN ANALOG MODULES	0		0		0	
ANALOG MODULES/UNIT	0		0		0	
COMP. IN DIGITAL MODULES	0		275	220 X 1.25	0	
DIGITAL MODULES/UNIT	0		10		0	
TYPES OF HYBRID MODULES	0		7		0	
SETS OF HYBRID MODULES	0		25		0	
BREADBOARD UNITS	1		1		1	
DEVELOPMENT UNITS	2		2		2	
PROTOTYPE UNITS	20		20		20	
SUBCONTRACT DEVELOPMENT, ODC, K\$	290		300		420	
PURCHASED PARTS ODC, K\$	250		248		224	
HOURLY RATE, \$	10.80		10.80		10.80	
MONTHS TO DEV. PEAK	15		15		15	
MONTHS TO ODC PEAK	12		12		12	
MONTHS TO END OF PROGRAM	30		30		30	
BURDEN, PERCENT	138		138		138	
MISC. ODC, PERCENT	2		2		2	
DRAWING FACTOR	144		144		144	
SUBSYSTEM DESIGN	20		20		20	

TABLE 40. INPUT DATA FOR ESTIMATING HARDWARE DEVELOPMENT COSTS FOR CONFIGURATIONS 4, 5 AND 6

(DATE 19 MAY 1976)	CONFIGURATION					
	4		5		6	
	EXPECTED	COMMENTS/RISK	EXPECTED	COMMENTS/RISK	EXPECTED	COMMENTS/RISK
SPECIFICATIONS	48	2 UNIT + 1 HARNESS + 2 BIU 7 CARDS X 2 = 14 + 29 IC'S	65	2 DP + 2 MP + 1 HARNESS + 2 BIU + 9 CARDS X 2 = 18 + 40 IC'S		2 DP + 2MP + 1 HARNESS + 2 BIU + 7 CARDS 55 X 2 = 14 + 34 IC'S
ANALOG STAGES	0		0		0	
DIGITAL COMPONENTS	115	12 + 103	255		190	
BREADBOARD ASSEMBLIES	4.2	211 X 2/100	10.2	508 X 2/100	7.4	370 X 2/100
EXPERIMENTAL DRAWINGS	83		115		97	
PRODUCTION DRAWINGS	83		115		97	
PARTS TO BE QUALIFIED	29		40		34	
NEW MATERIAL AND PROCESSES	2		2		2	
DEVELOPMENT TESTS	592		838		662	
ACCEPTANCE TESTS	1380		2100		1500	
DAYS OF STE	4		4		4	
DEVELOPMENT ASSEMBLIES	25.8		41.6		35.4	
PROTOTYPE ASSEMBLIES	258		416		354	
SUBCONTRACT SUPPORT, min	21		25		25	
TOTAL COMPONENTS	264	211 X 1.25	635	508 X 1.25	515	412 X 1.25
COMP. IN ANALOG MODULES	0		0		0	
ANALOG MODULES/UNIT	0		0		0	
COMP. IN DIGITAL MODULES	0		0		0	
DIGITAL MODULES/UNIT	0		0		0	
TYPES OF HYBRID MODULES	0		0		0	
SETS OF HYBRID MODULES	0		0		0	
BREADBOARD UNITS	1		1		1	
DEVELOPMENT UNITS	2		2		2	
PROTOTYPE UNITS	20		20		20	
SUBCONTRACT DEVELOP- MENT ODC, K\$	500		340		360	
PURCHASED PARTS ODC, K\$	208		286		317	
HOURLY RATE, \$	10.80		10.80		10.80	
MONTHS TO DEV. PEAK	15		15		15	
MONTHS TO ODC PEAK	12		12		12	
MONTHS TO END OF PROGRAM	30		30		30	
BURDEN, PERCENT	138		138		138	
MISC. ODC, PERCENT	2		2		2	
DRAWING FACTOR	144		144		144	
SUBSYSTEM DESIGN	20		20		20	



Configuration 5 is apparently related to the larger number of electronic parts in that design. However, since the larger number of parts is associated with the second processor (of identical design) it would seem that the development labor should not be much greater than in Configuration 1. Manufacturing, labor, and ODC would be greater, however. The entries in the table indicate that development labor is some 600 thousand dollars greater for 5 than for 1. This does seem extreme even though more testing is involved. It is possible that more refined input would cause the hardware development cost for Configuration 5 to come more in line with the other configurations. If so, one must conclude that variations in total program costs for engineering development of the six designs is not very significant.

#### 9.5.2 Risk Assessment

None of the designs is technically infeasible so the risk factor is primarily with respect to schedule. A 30-month schedule was assumed for all cases.

Configuration 2 is an existing, tested design. There is essentially no risk connected with this system. Where development risk might come in is if one desired to decrease power consumption by using low power Schottky chips to the greatest extent possible. The attempt to hold throughput constant while reducing power consumption poses a design problem that could threaten the schedule. The risk is not believed to be very great.

The major risk in Configuration 1 is that the LSI bipolar microprocessor chips have not been thoroughly characterized. If the chosen chips do not live up to their promise, a schedule delay might result because of the necessity for finding a substitute chip set or for design changes to accommodate the lesser performance of the chips. A similar conclusion holds for Configuration 5 although in this case throughput is not quite so important. Configurations 1, 2, and 5 are all considered to be in a low risk category.

There are two risk factors associated with Configuration 3. The first is the design of the LSI chip and the second is the design of the memory management circuitry which is a new design. Since an LSI design very similar to what is required already exists, the risk is not thought to be great. Considering the two risk factors, the risk for Configuration 3 is judged moderate.

The major question with respect to risk in Configuration 4 has to do with the schedule for the design of the LSI CPU. This is a new, rather demanding design. The probability of obtaining a successful design in much less than 2 years is rather small. Thus, considering the 30-month schedule, the risk would be rather high. For a 36-month schedule, the risk would be no greater than for Configuration 3.

The hardware risk factors for Configuration 6 are (1) CPU chips have not been characterized, therefore not proven; (2) memory management circuitry must be designed. These factors are not of too much

concern. From strictly this viewpoint, the risk is on the low side of moderate (less than Configuration 3). In this configuration, however, there is some question of suitability for the DP application.

### 9.5.3 Software Considerations

The two comparisons to be made are for single processors versus distributed processors and monoproductors versus multiprocessors. The factors to be considered are software development and software maintenance. There seems to be little or no hard data available that pertains to the situation.\* For example, there is a body of opinion that producing software for a multiprocessor is more difficult than for a monoproductor. However, no controlled experiments have been made and the literature does not seem to have any comparisons of the two cases. There is no doubt that the multiprocessor does introduce new difficulties into software design; particularly with respect to shared resources. While the guesses as to the increased difficulty have ranged above 100 percent, a more reasonable guess might be about 30 percent extra difficulty for development.

### Central Processor Versus Distributed Processor

If there is a convenient functional partitioning for the total set of tasks, the software development effort should be about the same for both configurations. Where a difference might arise is with a task, not easily partitioned, which must be shared between two processors. This kind of situation might arise when a new task is added to the system and while the remaining growth capability is adequate, it is scattered among two or more processors. In other words, it is more difficult to efficiently use the growth capacity in the distributed processor case. Another point of difference has to do with maintenance. Any change which affects both system integration functions and a subsystem function will require a memory change in two or more processors in the distributed case versus one in the central case. This would be true, for example, if the autopilot function was in a different processor than the integration function and a change were made to the system that necessitated an autopilot change.

---

\*Available data about commercial large-scale data processing systems are not felt to be applicable to the real-time missile processor software requirement.

# INITIAL DISTRIBUTION

Hq USAF/RDPA	1	CG USAMICOM/AMCPM-CT-E	1
Hq USAF/RDQRME	1	DDC	2
Hq USAF/XOOF	1	AFATL/DLJA	1
AFSC/INA	1	AFATL/DLJF	1
AFSC/SDA	1	USAF Academy	1
AFSC/DLCAW	1	ASD/SD-65	1
ASD/YFEI	1	AUL (AUL-LSE-70-239)	1
ASD/YFEA	1	ASD/ENFEA	1
ASD/YPEX	1	Hq USAF/SAMI	1
ASD/SD	1	Ogden ALC/MMM	2
AFFDL/FEI	1	AFIS/INTA	1
AFAL/RW	2	Hq TAC/DRA	1
ASD/ENA	3	Hq USAFE/DOQ	1
TAC/DRAI	2	Hq PACAF/DOO	1
TAC/XPSY	1		
AFAL/AAI	1		
ASD/YHEV	1		
ASD/XRG	2		
NAVAIR SYS COMD/AIR 360E	1		
NWC/Code 143	4		
NWC/Code 533	1		
AFFDL/FGL	1		
ATC (XPQS)	1		
NAVAIR SYS COMD/AIR-5323	1		
NAVAIR SYS COMD/AIR-5324	1		
OSD, ODDR&E/TST&E	1		
DARPA/TIO	1		
ADTC/PP	1		
ADTC/ADE	1		
TAWC/DT	1		
TAWC/TEFA	1		
TAWC/FTS	1		
ADTC/XR	2		
TRADDC/ADTC/DO	1		
AFATL/DL	1		
AFATL/DLB	1		
AFATL/DLA	1		
AFATL/DLMA	1		
ADTC/SD15	1		
AFATL/DLMT	1		
AFATL/DLMM	15		
AFATL/DLY	2		
ADTC/SD-7	1		
TAWC/TRADOCLO	1		
AFATL/DLOSL	9		
Redstone Sci Info Center	2		
Off Naval Research/Code 121	1		